International Journal of Advanced Research in

Computer Networking,Wireless and Mobile Communications

Volume: 2 Issue: 1 11-Apr-2015,ISSN_NO: 2320-7248

Friend Recommending Peer to Peer File Sharing And Synchronization Application

Kharade Varsha, Jagtap Sandhyarani, Pawar Yojana, More Umesh . Guide : Prof. Belsare P. P.

Department Of Computer Engineering, SBPCOE, Indapur (Pune)

Abstract-*P2P architecture is the next-generation network paradigm to replace the traditional client-server architecture. Typical P2P systems are characterized by the decentralized control, scalability and robustness. Centrally managed storage services, such as Dropbox, are popular for synchronizing data between several devices. P2P-based approaches that run fully decentralized, such as BitTorrent-Sync,are starting to emerge. This paper presents Box2Box, a new P2P file synchronization application which supports novel features not present in BitTorrent-Sync.*

Keywords:

Peer-to-peer, Synchronization, Bit-torrent, Friend Recommendation, File Sharing.

1. Introduction

Peer-to-peer is a decentralized communications model in which each party has the same capabilities and either party can initiate a communication session. Unlike the client/server model, in which the client makes a service request and the server fulfills the request, the P2P model allows each node in a peer-to-peer network to function as both a client and a server. Two computers are considered peers if they are communicating with each other and playing similar roles. End users in a P2P network must first download and execute a peer-to-peer networking program. After launching the program, the user then enters or selects the address of another computer belonging to the network. The address, which may look like a screen name or virtual phone number, is actually an IP address. BitTorrent is a way to transfer files of just about any size quickly and efficiently. It works by breaking files up into small pieces. The file is downloaded piece by piece from one or many different sources. It's efficient because you get faster downloads using a lot less bandwidth. The name BitTorrent is also used to describe the official BitTorrent client

2, System Features

2.1 Mandatory Features

Features, that were required and are successfully implemented in our system include:

• user interface: we created a fully functional GUI on top of our application that allows to manage all of the core functions.

• robustness: our system handles churn and fully recovers from a connection loss of any peer.

• pure P2P: the system does not rely on any centralized entity. It is designed to be fully distributed .

• license free: any included library or framework is free for use or licensed under a GPL license.

• multiple machines: our systems supports the usage of multiple machines for the same user. A registered user is able to synchronize her own data on any device, even if the original source is currently offline.

• file sharing with friends: we maintain a list of friends and pending requests. Users can share data on per-file or per-folder basis with a arbitrary range of friends.

• conflict management: conflicts are recognized, marked and conflict copies maintained. It is then the user's choice how to resolve occurring conflicts.

• friend recommendation: new friends can be suggested based on a network of common friends. The recommendation is ranked according to relevance .

2.2 Additional Features

We decided to implement the following features, although they were not explicitly requested:

• *versioning*: users can revert changes in documents in order to reset the content to a previous version. There is a preview function in place, so that users can check the older versions before reverting the current one. After the reset, the operation is synchronized to other users who share the file.

• *authentication*: to restrict the access to sensitive data, the application is password protected.

• *encryption*: the connections and shared data between users are 2048bit encrypted to provide maximum confidentiality.

• *change recognition*: our system relieves the user from any responsibility to track changes in files manually. Every change from both user and friends are detected and synchronized fully automated.

3. Techniques

The chocking algorithm is the core of the BitTorrent protocol. The following sections explain the algorithm and the main concepts behind the BitTorrent protocol.

International Journal of Advanced Research in

Computer Networking,Wireless and Mobile Communications

Volume: 2 Issue: 1 11-Apr-2015,ISSN_NO: 2320-7248

3.1 The choking algorithm

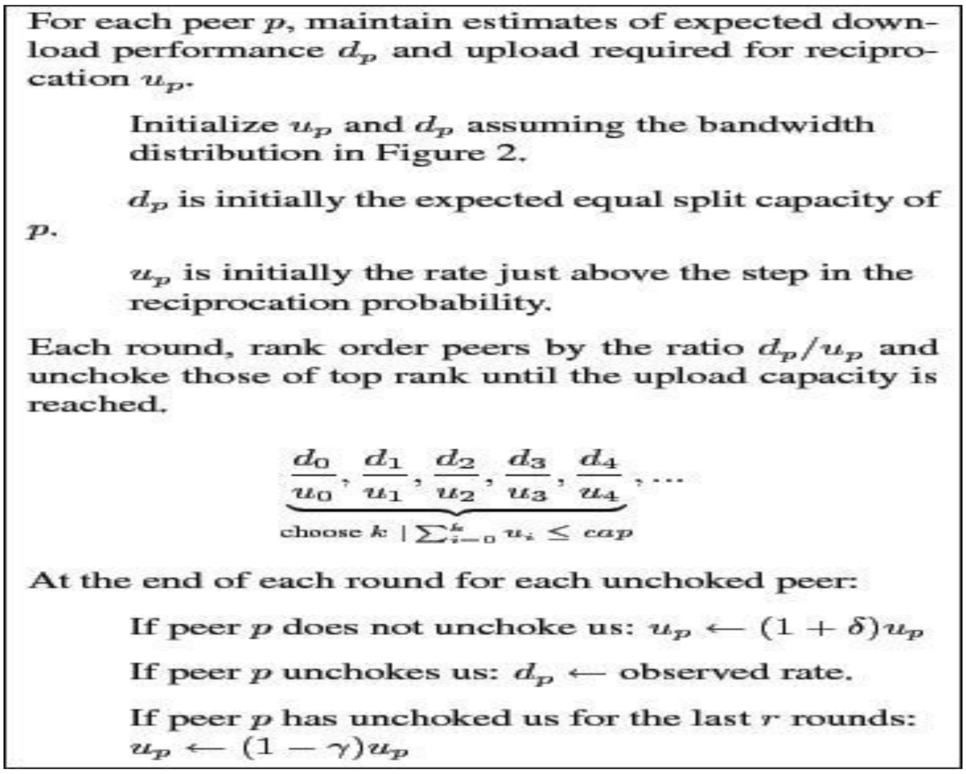
The general algorithm behind BitTorrent clients is commonly referred as the choking algorithm, that, together with a variant of tit-for-tat mechanism is the basis for all the different clients that use the protocol. In the choking algorithm, every peer is responsible for choosing its own local neighbourhood.

The trackers are responsible for the coordination among peers, they keep track of the information shared through the swarm. A number of notifications are sent periodically to the tracker from every client, regarding pieces of the file that the clients possess and pieces that they request.

When uploading, leechers prefer peers that have better upload rates, but seeders always perform a round robin unchoke scheme. Leechers periodically check other peers in the neighbourhood. If there is a large number of peers which ask for uploading, then the leechers choose (or unchoke) a certain number of peers to who upload to.

3.2 The optimistic unchoke

The optimistic unchoking in one of the most important aspect of BitTorrent protocol. All peers perform optimistic unchoke, which means that every peer, randomly, chooses another peer in the swarm and upload data to it. The choice is repeated each 30 seconds. During this time interval an user is able to connect to other peer with a medium-high download rate.



For each peer p, maintain estimates of expected download performance d_p and upload required for reciprocation u_p.

Initialize u_p and d_p assuming the bandwidth distribution in Figure 2.

d_p is initially the expected equal split capacity of p.

u_p is initially the rate just above the step in the reciprocation probability.

Each round, rank order peers by the ratio d_p/u_p and unchoke those of top rank until the upload capacity is reached.

$$\underbrace{\frac{d_0}{u_0}, \frac{d_1}{u_1}, \frac{d_2}{u_2}, \frac{d_3}{u_3}, \frac{d_4}{u_4}}_{\text{choose } k \,\mid\, \sum_{i=0}^{k} u_i \leq cap}, \ldots$$

At the end of each round for each unchoked peer:

If peer p does not unchoke us: $u_p \leftarrow (1 + \delta) u_p$

If peer p unchokes us: $d_p \leftarrow$ observed rate.

If peer p has unchoked us for the last r rounds: $u_p \leftarrow (1 - \gamma) u_p$

This mechanism is important for two reasons. First, it allows users to enter the swarm, because in a tit-for-tat system, a peer needs to have some data before it can share blocks of file. Second, it helps peers to find a faster active set, improving their download speed.

3.3 The tit-for-tat mechanism

The tit-for-tat mechanism is used directly between leechers to exchange data in a fast and fair way. In this mechanism peers provide blocks to those who have provided them blocks in the past. Leechers are interested in peers which have the parts of the file that they miss. That is why the optimistic unchoke is necessary. A new leecher will never be seen as interesting if it does not possess any part of the file. And it will not exchange information.

Each clients performs an optimistic unchoke every 30 seconds. The goal for high efficiency is then set up several connections that transfer data in both directions.

4, Design and Implementation structure

4.1 P2P Network

The complete communication between any two computers running Box2Box is handled over a Peer-to-peer network . That means that every computer is registered as a peer in a P2P network and queries this network for data, information or important updates. There are no direct network connections.

4.2 P2P Abstraction Layer

As a library, which enables us to build and maintain a fully-etched Distributed Hash Table (DHT in short) P2P network, TomP2P is used. However, in order other components in Box2Box can easily and reliably access the generic methods in the TomP2P library, we designed and implemented a P2P Abstraction Layer (fig.). Besides translating the domain-specific method calls in Box2Box to the existing generic methods in TomP2P, other important tasks of the abstraction layer are to set up and maintain a TomP2P Peer instance while run-time, keep track of registering the user and the computer in the P2P network and make use of a Transparent Encryption Layer.

4.3 P2P Sharing and Querying Process

As already described, the P2P Abstraction Layer covers various communication aspects. Many different data, information and status can be written into and queried from the P2P network. First, a particular document A is shared by one node in the P2P network. Since TomP2P is an implementation of a DHT, data has to be stored as key-value pairs. In our example, the document identifier as well as the timestamp of the document build together the key, whereas the document itself is the data In the second half of the exemplarily sharing and quering process, another document B is queried from the P2P network.

4.4 GUI

The GUI part of the system is implemented with JavaFX and initialized in the main method of GuiMain.s We decided to include means for every system function in the graphical interface, in order to provide the user with a more intuitive mode of operation compared to command-line interaction. The resulting interface is limited by only two constraints. Firstly, the preview operation to open any of the listed documents in the default application is not supported by all platforms and therefore only available in Linux and Windows.

4.5 Synchronization

The here presented system is based on the principle of registering a dedicated file system directory, we called it Box2Box directory, for tracking file addition, removal and modification. File synchronization of a users documents and documents shared among users is accomplished with a serializable Java objects of type DocumentsRegistry. Complementarily, the interface FileSynchronizer provides the necessary methods to forward local fle system changes to the relevant other peers in the network. FileSyncObservable and FileSynchronizer are both implemented in the class FileSynchronizer.

4.6 Friendship Recommendation

The recommender system to generate friendship suggestions is based on the relationship graph of the user. If a user requests new recommendations the graph is constructed up to a specified depth, in which each of the nodes correspond to a user and represents a possible friendship recommendation.

4.7 Encryption

One interesting feature of Box2Box is its ability to transparently encrypt and decrypt while communicating over the P2P network. A user normally does not notice the Encryption subsystem which is working behind the scenes. This is realized by the concept of shared key rings. The main idea behind shared key rings as well as the different key types used by Box2Box is now further discussed in this section.

4.8 Different Key Types

In order the encryption is independent from any computer the user is currently logged in,all keys needed by the peers have to be available in the P2P network. In the long run, this means that all keys ever generated or used have to be shared in the P2P network. The following list describes briefly these four key types.

Password Key: The password key is generated after the user has logged in with his user identification and his password. Out of this password, which is a string as data type, a symmetric key is generated. The point of this key is that for the same string always the same symmetric key is generated.

- **Private Key:**As part of a key pair, the private key is generated while registering using the user identification. Obviously, the private key and public key is bound to the user and his identification. Both keys are then shared in the P2P network and also stored on the disk of any computer, on which the user has logged in.
- **Public Key:**Since the public key is part of the key pair, it is generated during user registration, too. For all four key types, the public key is the easiest case because it never has to be encrypted can be freely shared in the P2P network and can also be written on disk without any further security measurements.
- **Document Key:**The document key is again a symmetric key and it is generated in the situations when a new friend is added to the friend list of the user. The reason for this is that a document key is bound to two particular users the sender and the receiver of a document.

5. DEMO SCENARIO

The scenario is based on one user running three peers on three devices, two peers on a laptop (P1 and P2) and one peer on a UNaDa (P3). In addition, a large network of 100 peers is running in the background on a single laptop. In this scenario the demo presents five use cases. These use cases are depicted in Figure 1 and show the setup of the demonstration and the peer interaction in each use case. The gray box that includes P1, P2, and P3 represents one user. The peers P4-P104 have one user per peer, resulting in 100 users. The GUI and console log of P1 and P2 will be shown on screens.

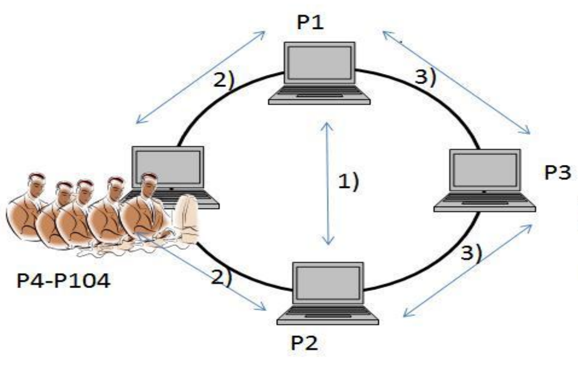


Fig.1(a) Use Cases 1-3

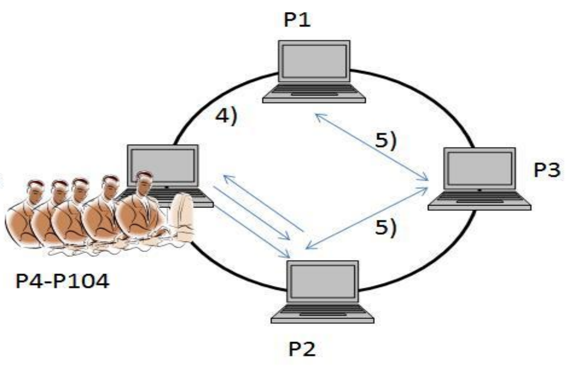


Fig.1(b) Use Cases 4-5

1) P1 is online and P2 is offline. The user, on P1, adds a large file to B2B, since P2 is offline no synchronization is possible. P1 goes offline and P2 comes online. Still, no synchronization will happen, since no direct transfer is possible. P2 will be notified about the new file though. P1 comes online again and the large file is transferred to P2, similar to BitTorrent-Sync (cf. Figure 1a). The synchronized file appears on P2 and the synchronization process can be observed in the log files.

2) Only P1 is online to the network and adds a small file to B2B. This file is encrypted and stored in the underlying P2P-network (P1 - P104). P1 goes offline and P2 comes online. Because the small file is stored in the network it can be transferred to P2 without P1 being online. herefore, P2 downloads and decrypts the file (cf. Figure 1a).

3) The user installs Box2Box on his UNaDa (P3), actinas the super peer which stores all files (cf. Figure 1a). This means that all the files of the user are directly stored on the super peer.

4) The user on P2 requests a friend recommendation and receives a list of firends of friends. After the two users have mutually established their friend relation they can share files. The user on P2 shares a file with the selected friend. Since it is a small file it can be synchronized offline (cf. Figure 1b).

5) The two peers P1 and P2 update a file at the same time resulting in a conflict. The peer that was first to upload his file is continuing as usual. The Peer that was second receives a conflict notification (cf. Figure 1b). After the user resolves the conflict a new version of the file is created and uploaded.

6. System Architecture

In order to not only manage the complexity of such a fully distributed P2P software system,but also to keep any unwanted dependencies between different subsystems low,we implemented various subsystems (or modules) with isolated responsibilities. This approach also helped us in meeting the diverse system requirements imposed on Box2Bogx. An overview of the system architecture of the presented application is depicted in fig.

Basically, the system can be divided into the core system and the peripheral system. The core system consists of the P2P Abstraction Layer, the Encryption, the Community Interaction, the File Synchronizer and the File System Observer. As for the peripheral system, it includes the TomP2P library TomP2P1, the Graphical User Interface and the File System. Any subsystem can then be assigned to one of four abstraction layers. For the network communication, we make use of the TomP2P implementation of a Distributed Hash Table (DHT in short). The third abstraction layer contains the Community Interaction module, File Synchronizer module and the File System Observer module. These modules or software entities process all events created by any remote or local system state change.

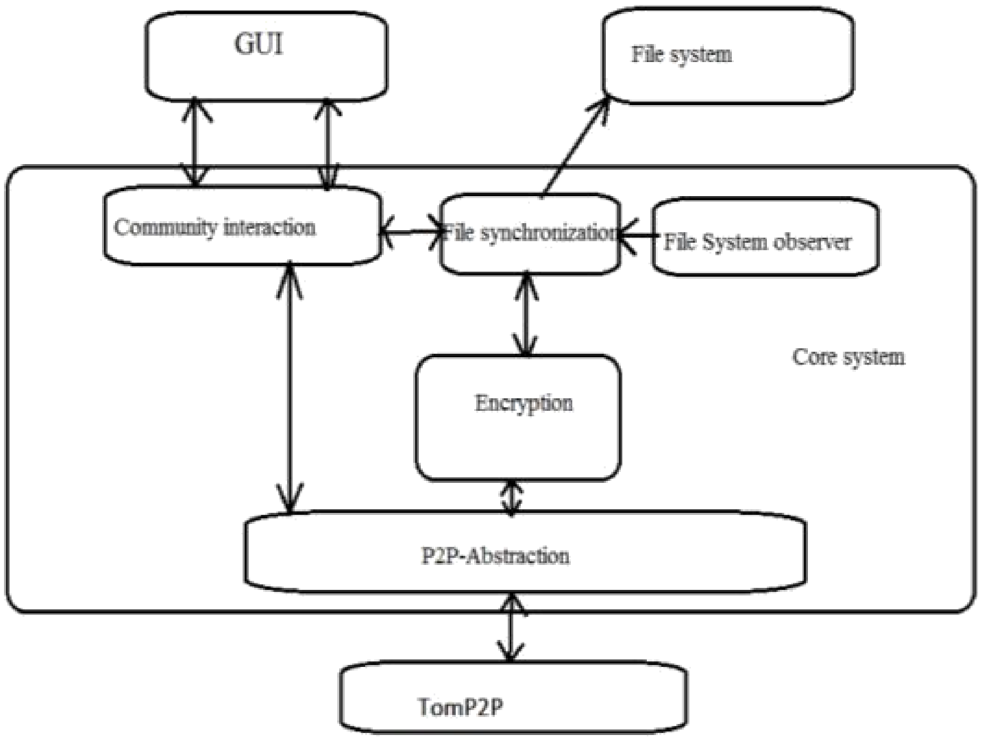


Fig:System Architecture

7. FUTURE WORK

The next step is to release Box2Box to the public as open source software. For the initial release several improvements are still necessary. Future work includes backup storage on friend peers. The user will be able to set the redundancy ratio and split the backup in a way that allows to reconstruct the backup if a given fraction of the friends are online (e.g., backup can be reconstructed if 3 of 5 friends are online).

Currently a JavaFX front-end is used, which will be replaced by a web-based GUI that is work in progress. For future work, a desktop integration that allows the user to use Box2Box transparently is foreseen. Besides encryption, future work investigates more security aspects and potential attack scenarios, such as access restriction or colluding peers. An interesting security aspect is to consider friend peers as trusted entities. Relying on these trusted peers can mitigate many attack scenarios.

References

[1]SandvineInc.UCL,“GlobalInternetPhenomenaReport1H”, http://www.sandvine.com/downloads/documents/Phenomena 1H 2013/Sandvine Global Internet Phenomena Report 1H 2013.pdf, last visited: 30.5.2013, May 2013.

[2] CISCO. (2013, Jan.) Cisco Visual Networking Index: Forecast and Methodology, 2011-2016. [Online].Available:http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white paper c11-481360 ns827 Networking Solutions White Paper.html

[3] B. Cohen, “Incentives Build Robustness in BitTorrent,” in 1st Workshop on Economics of Peer-to-Peer Systems (P2PECON), Berkeley, CA, USA, June 2003.

[4]MiniwattsMarketingGroup,“InternetGrowthStatistics,” http://www.internetworldstats.com/emarketing.htm, last visited: 5.8.2009, January 2008.

[5] Dropbox Inc. (2013, May) Dropbox. [Online]. Available: http://www.dropbox.com/

[6] Google Inc. (2013, May) Google drive. [Online]. Available: http://drive.google.com

[7] BBC. (2013, May) Megaupload file-sharing site shut down. [Online].Available: http://www.bbc.co.uk/news/technology-16642369