FaceBook-based
Access controls for File sharing

Ambuj Thacker
Navdeep Bhatia
Vijit Singh Kharbanda

Department of Computer Science,
Stony Brook University
Stony Brook, NY, USA

Abstract

DirectConnect is a peer-to-peer file sharing protocol. The users are registered to a hub which initiates the peer to peer connection. All the users who are registered with the hub can share files with each other. There is a chance that a user does not want to share his files with every one registered on the hub, but only with his friends. Our goal in this project is to use social networking site (Facebook in our case) to allow file sharing only among the friends. We have designed, implemented and tested a scheme for using FaceBook friend's information to configure access controls using Advance Direct Connect (ADC) protocol.

1. Introduction

File sharing is a common way to exchange data or information between computer users. The peer to peer file sharing involves agreement between two users over sharing the file. Advance Direct Connect is a protocol which facilitates this type of file sharing. Users who want to share files should register themselves in the hub. The file requests are sent to the hub and the hub directs the user to the actual location of the file.

In the present scenario, administrator of the hub decides who can be a part of the complete network. Users do not have control over restricting other users over file sharing. For example: UserA want to share file with UserB but not with UserC (as they do not know each other). There are few public hubs in which every user can access even if they have not ever met. The users who become member of such hubs are aware of this fact. They then (we assume) share files accordingly.

One way to find friend`s of a user is to use his social networking account. In our project we have used Facebook to get the friend`s list of a user. Facebook is a Social Networking site that allows people to share information about them. We have used Advance Direct Connect Protocol for P2P

based file sharing system. Facebook based File Sharing system uses the friend list from Facebook and P2P sharing protocol from ADC to create access policies

1.1 Advance Direct Connect Protocol

Advanced Direct Connect (ADC) is a client/server-oriented messaging and files sharing protocol. ADC is the next generation of Direct Connect protocol. It is based on the Direct Connect topology. ADC clients connect to a hub. The hub stores the information about the file location. Using this location information, users can search for files and download them from other clients. The protocol also has the facility for users to chat with other users. All the commands are sent as plain text except during the password authentication. The hub uses the 411 port and all the clients connect to this port. There is no specific port for the clients.

The following is the protocol to download the file from

- All messages begin with a four-letter word. The first letter designates how the message should be sent and the other three specify what to do.
- Parameters are separated by space
- All text must be sent as UTF-8 encoded Unicode.
- Client and hub ignores unknown formatted messages.
- Client addresses is the IP address. Hub addresses must be specified in URL form, (adc://server:411/).
- Numbers are sent as strings. Client and hub should be implemented in such a way that it can handle such strings
- SIDs, PIDs, CIDs, and short binary data are sent as base32-encoded strings.
- ADC is case-sensitive. It uses upper case alphabets for command names.
- Some commands and functionality require the use of a hash function. The hash function is decided upon during the session setup and does not change till the session ends.

Users are identified with their nickname on a hub. Different hubs may have different nick name for the same user. While establishing connection between clients, a random number is exchanged between them. This random number is used to prioritize the download. The client with the highest number is allowed to download first. Downloads are transported using TCP. The connection to the hub is with TCP. A user can be in, either of the two modes supported by the protocol, "active" or "passive" mode. Clients using active mode can download from anyone else on the network. Clients using passive mode users can only download from active users.

Each client is identified by three IDs, namely Session ID (SID), Private ID (PID), and Client ID (CID). Session ID is a unique identifier given to user. It is assigned by the hub during initial protocol. Private ID identifies the client globally. They are used to generate CID. Client IDs globally and publicly identify a unique client and are used in client to client communication.

In this paper, we explain the design (section 2) and implementation (section 3) of the application along with the network call flows (section 4) for better understanding of the overall design. Section 5 depicts the screen shots and section 6 describes the problems encountered and the

design decisions taken to overcome them. We conclude (section 9) the paper by presenting the limitations (section 7), future work (section 8) and references (section 10).

2. Design

The Idea behind FB based file sharing is to import the friend list from user's facebook account and construct the Access Control based on the friend list and the policy configured. This ACL will then be used to grant/reject permission for P2P file sharing.

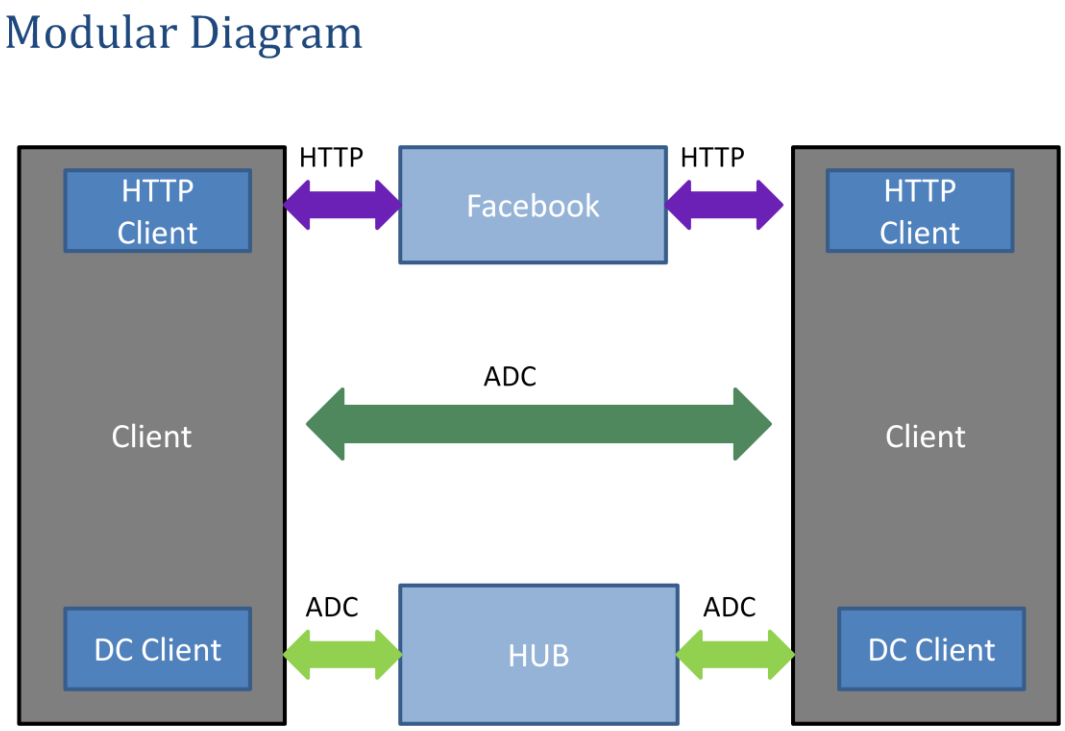

Figure 1: High Level Component Diagram of FB-DC Application

Every user has two clients running on their system (1) HTTP-Client and (2) DC-Client. The HTTP-Client is responsible to fetch the user`s friend list from Facebook. It connects to the Facebook, authenticates the user, and returns the complete list of friends. The DC-Client handles the functionality of communicating with HUB as well as the peer client based on ADC protocol.

2.1 The Basic Idea

The design of the application is based on the fact that service provider can control its privacy settings but cannot control subscriber's settings. In the case of facebook based access control, the client requesting the file is subscriber and the one who is the owner is the service provider. When subscriber requests the service provider, it also sends its own facebook UID. Service provider

based on its policy checks whether the given person is his friend or not. If yes, it writes a random number on its wall which can be read only by its friends. Service provider then sends a verify request to the subscriber. In response to the verify request, subscriber reads the random number from the service provider's wall and sends it across. On receiving the token, the service provider verifies the token and accordingly proceeds/rejects the download request.

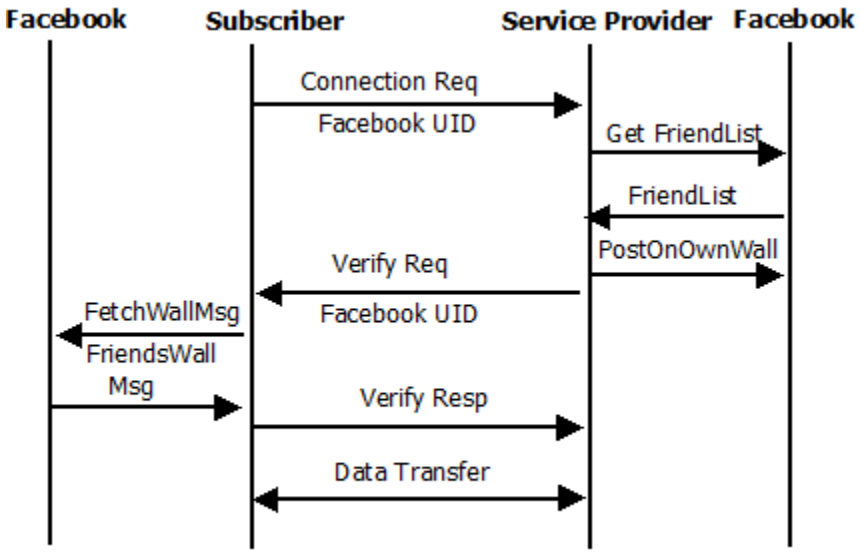

Figure 2: Call Flow for Policy "Friends"

3. Implementation

The implementation can be divided into two categories

1) Facebook Related
2) DC Client Related

The hub was used as is.

3.1 Facebook Authentication and Fetching data

Facebook provides multiple set of APIs to communicate with it to fetch data and perform various operations. The program uses REST APIs to get the friend list, post the message on the wall and uses FQL to fetch the wall message from the friend's wall. The program also uses Facebook as a web-service to authenticate the users using the OAuth 2.0 protocol. It uses the login page provided by Facebook and sends the information to the Facebook over https and asks for offline access. On getting the offline access, the program uses the access token generated by the authentication for further communication with the Facebook. All this is done via an application for Facebook. The data received from Facebook is in json format.

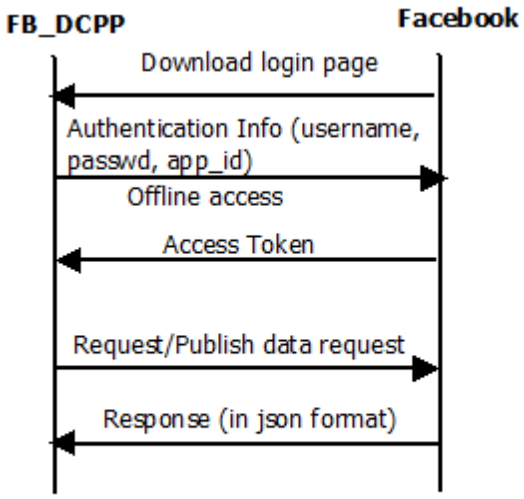

Figure 3: Facebook - HTTP Client Communication

3.2 Direct Connect Client Changes

The Direct Connect client based on ADC protocol was modified for the following operations:

1) Connection Request was updated to contain Facebook UID apart from nick.
2) Additional handling on receiving the connection request was introduced to perform authentication based on the Policy enforced.
3) New messages Verify request and response were introduced to carry service provider's UID and verification token respectively.
4) State machine was updated to contain new State STATE_VERIFY to perform the verification before proceeding to the file transfer state.

3.3 Policy Settings

Currently only two policy settings are allowed:

1) ALL
2) Friends

4 Call Flows

4.1 Policy setting "friends"

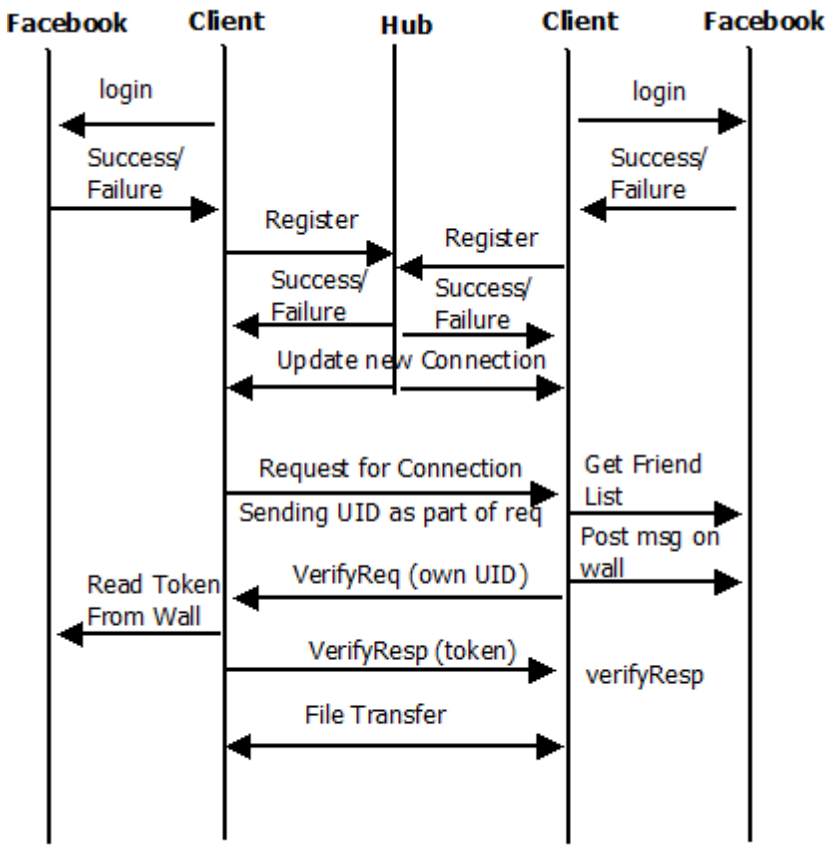

Figure 4: Network Call Flow for Successful Scenario

File sharing between Client A and Client B (has the actual file):

Step 1: Clients log into their facebook account using the HTTP-Client. The client gets denied or successful login based on his Facebook authentication.

Step 2: Clients (already registered with the hub) log into the hub to share file. Upon successful login the users can request for file transfer. Each client gets the list of users who are registered with the hub.

Step 3: Client-A requests Client-B for connection and sends his UID.

Step 4: Client-B on receiving the request gets friend list from facebook and checks if the UID received is the UID of a friend on Facebook.

> *Step 4.1*: If UID received is the UID of one of his friend on Facebook, it posts a message on his own wall. The privacy settings on Facebook for this client are such that only his friends can read the wall.
>
> *Step 4.2*: After this, Cleint-B sends his own UID to Client-A.

*S*tep 4.3 Client A then reads Cleint-B facebook wall and sends him the token written on the wall. If the requesting client responds with the correct token , then the other client-B is assured that the person at the other end is his friend and a File Transfer follows.

Using the above call log the users are ensured that the files they share is with the users they know. This provides additional security to the current ADC file sharing methodology.

When Requesting Client A is not a friend of client B

If Client A requests access to file from Client B, Client A sends his Facebook ID to Client B. Client B gets his Friends List from Facebook and checks if any of his friends has that ID. In case, none of the friend of Client B has that Facebook ID, a failure message is sent.

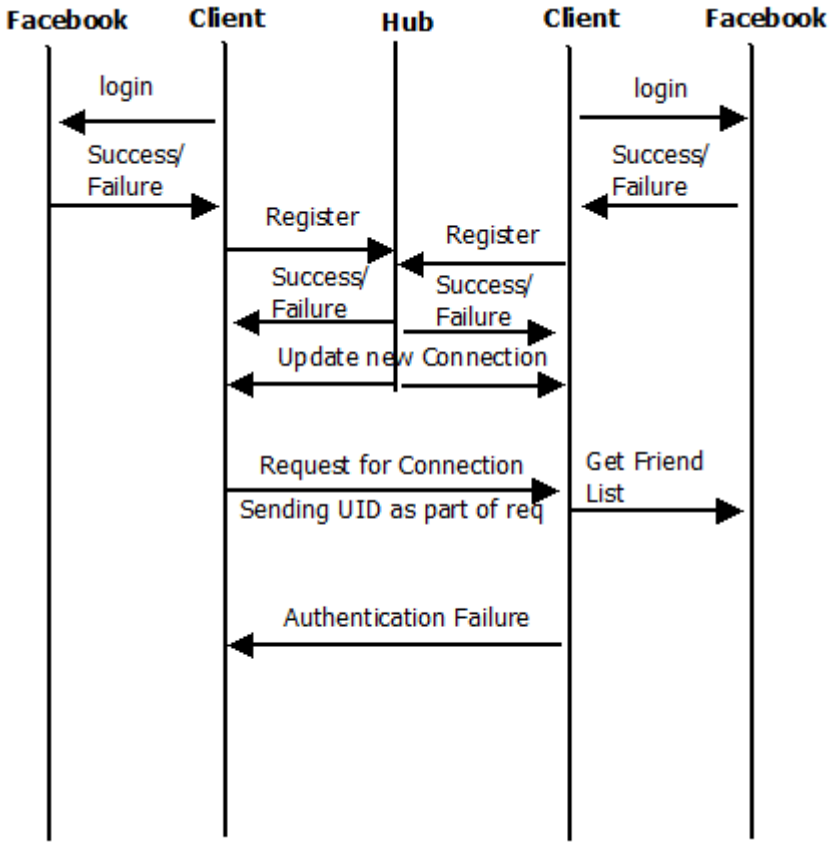

Figure 5: Network Call Flow for User Authentication Failure Scenario

When Client A pretends to be a friend of Client B

If Clients A pretends to be a friend of Client B by sending the UID of some friend of Client B , Cleint A would not be able to read the wall of client B and hence would fail to provide correct token to client when requested to do so.

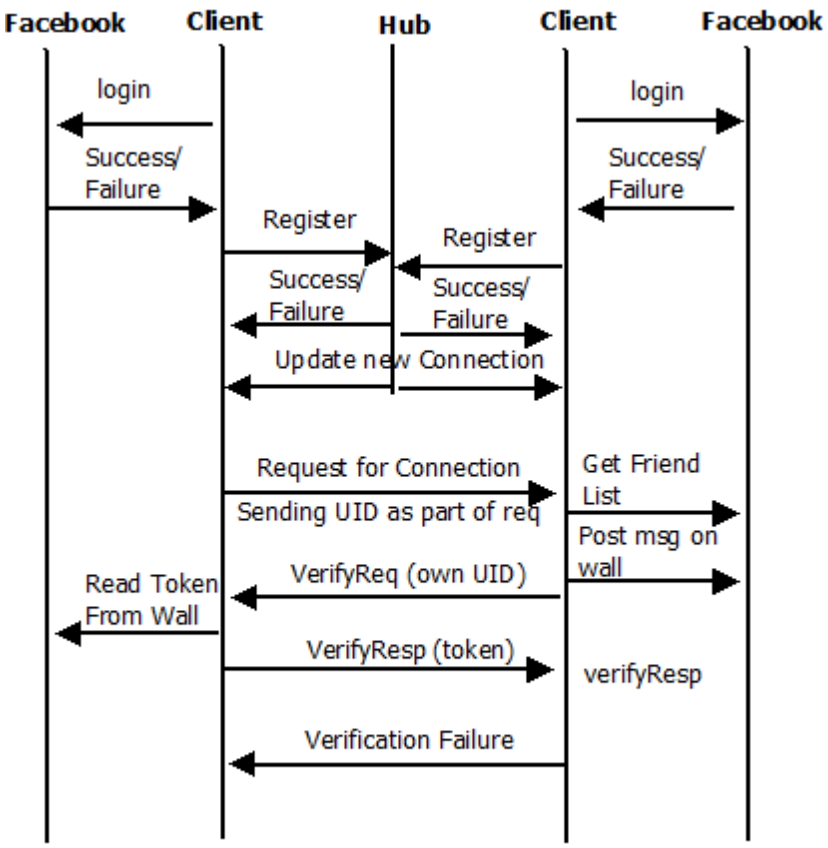

Figure 6: Network Flow for Verification Failure Scenario

4.2 Policy setting "ALL"

Client A sends a request for a connection along with his UID but client B doesn't checks if the Received UID is actually the UID of one of his friend. We haven't changed the state machine for this policy , hence Client B still posts a message on wall but doesn't verifies the token sent by client A and allows File transfer to happen.

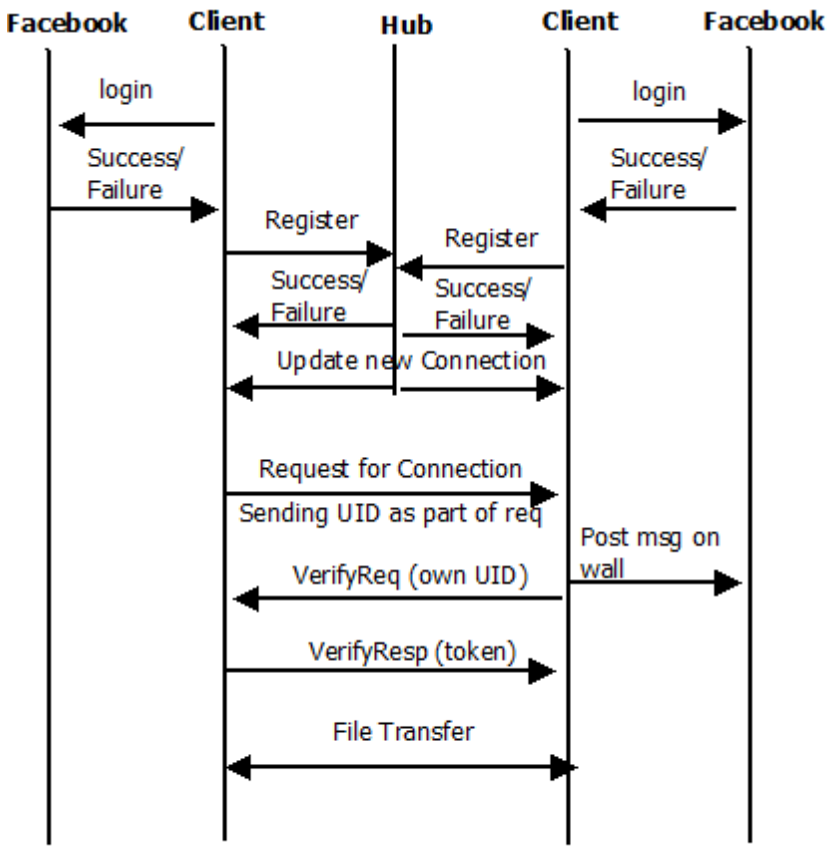

Figure 7: Network Flow for Policy ALL

5 Screen Shots

The following section depicts few of the screens that users see on running the application:

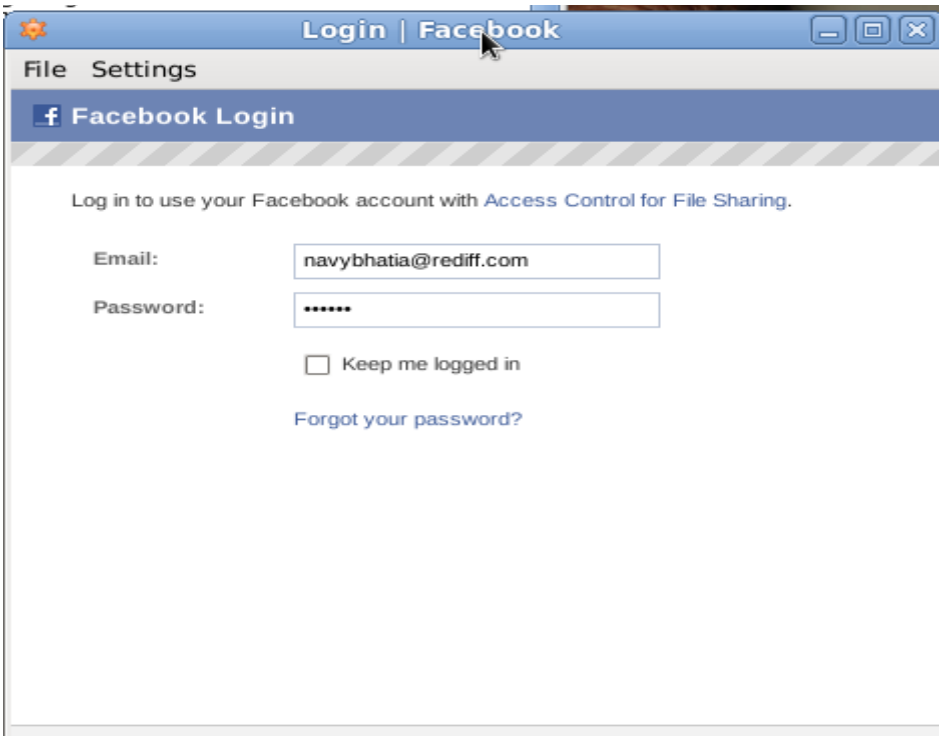

Figure 8: Facebook Login Page

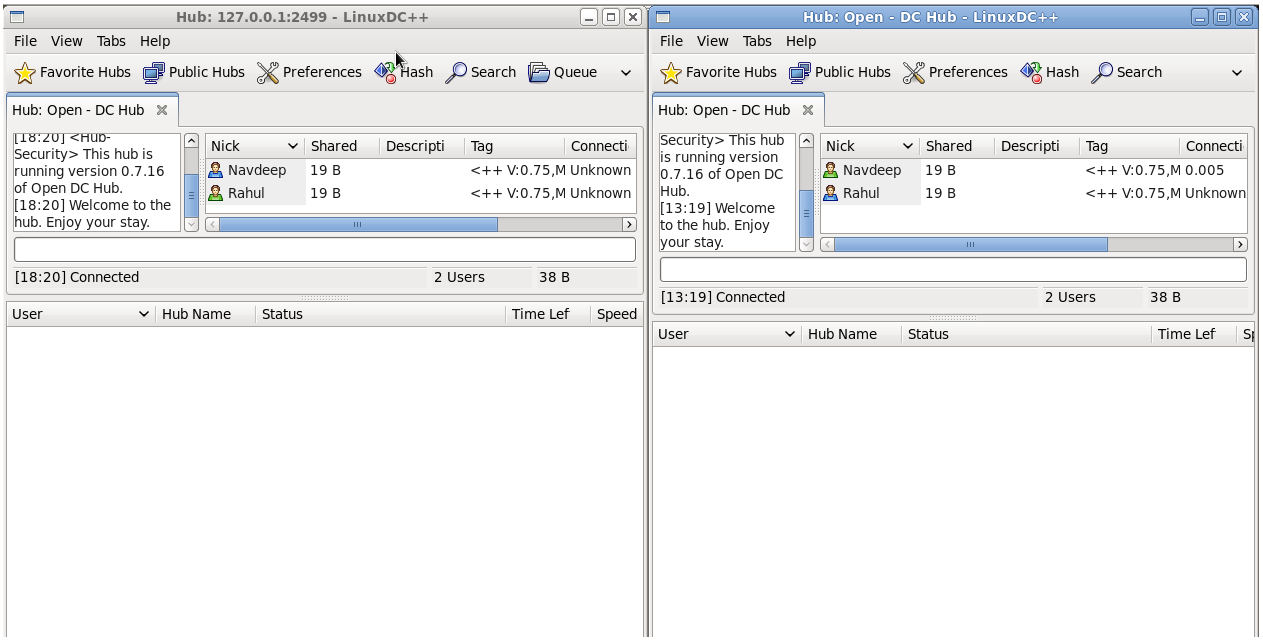

Figure 9: Direct Connect Client Screen after connecting to HUB

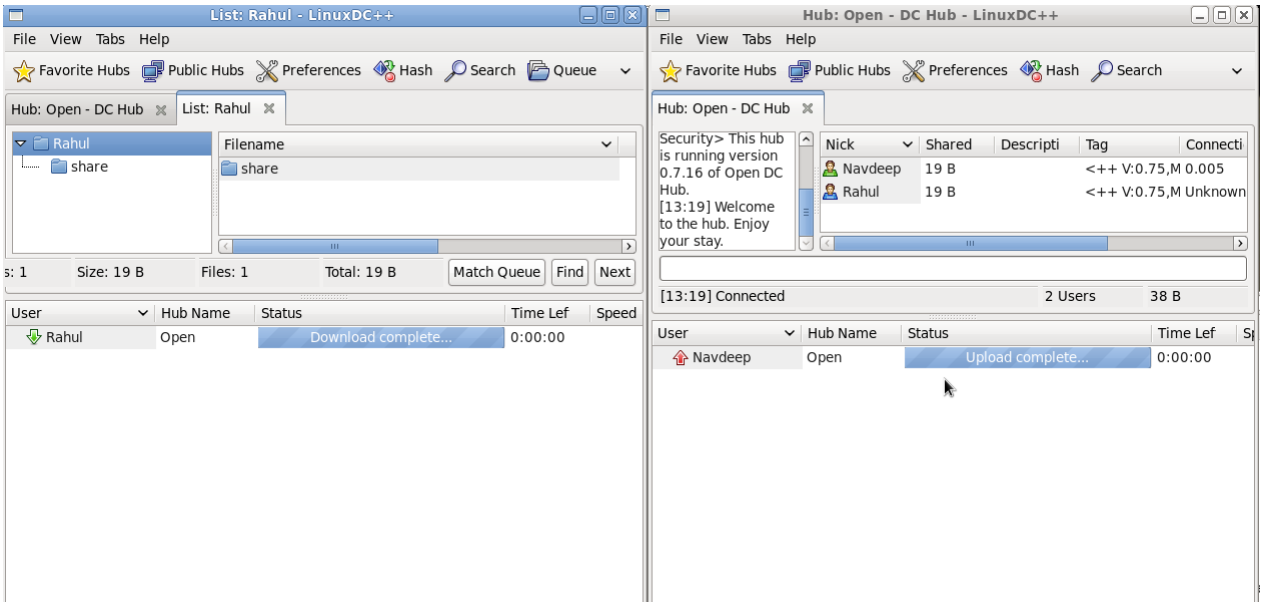

Figure 10: Direct Connect Client Screens during File Transfer

6. Design Decisions/Problems Encountered

- Single Executable: Our initial approach was to use C# as the HTTP-Client. We were successful in generating the friend list from the Facebook. Due to integration issues with the present ADC clients we then moved to create a separate application which can then communicate to any client using socket communication. This would have caused performance issues. We intended to create a single executable to make our end implementation an easy-to-use application. The idea of creating a single executable led us to merge the HTTP Client and DC client and they communicate with each other via synchronous calls.

- Facebook Provided APIs (REST APIs) [1] for Data Extraction and Publishing- the REST API enables you to interact with Facebook web site programmatically via HTTP requests. The issue with the REST API was that you can only use it to read your own wall and not your friends. Therefore, we use FQL to read token from the wall.

- Using Exiting web service provided by Facebook.

- OAuth 2.0 protocol for authentication and authorization for Facebook.

- Facebook deprecated sending private messages to users to get rid of spam and therefore wall was used to post messages. (http://developers.facebook.com/docs/reference/rest/sms.send)

- Facebook does not allow extraction of Privacy settings so that cannot be used**.**

- Policy Configuration: Users can define policies to construct ACL along with the Facebook groups and networks in their policies. Currently supported policies are as follows:
    - ALL - This policy implies that everyone would be able to access the file.
    - FRIENDS - This policy implies that only the friends present in the Facebook Friend List would be able to access the file.

7. Limitations

- Facebook does not allow sending private messages (API deprecated) so the idea was implemented by posting tokens on wall that was only accessible by friends. The security can further be strengthened by sending email but that would require human intervention.

- Current Implementation only authenticates that the client requesting the file is one of the friend of the client owning the file. So his friend can pretend as one of his another friend and still would be successful in retrieving the file.

8. Future Work

The application can be extended to support various features such as:

- Support for ACL based on Specific Friend lists can be provided.
- Support for ACL based on Specific Groups can also be provided.
- Run time (and not init time) policy change support.

9. Conclusion

We have successfully implemented Facebook based access control for file sharing. The users connected to a hub can now choose to share files with their friends. We have also given the choice to the user to share the file with ‘every one’. This way our implementation can behave in the old way of allowing every hub registered user to share files among themselves. Other social networking sites can also be used to derive friend list. Incorporating new social networking websites will involve modifying the HTTP_Client only.

10. References

[1] Facebook REST API
http://developers.facebook.com/docs/reference/rest/

[2] Facebook Developer Documentation
http://developers.facebook.com/docs/

[3] Direct Connect Client/Hub
http://dcplusplus.sourceforge.net/
http://opendchub.sourceforge.net/

[4] ADC Protocol
http://adc.sourceforge.net/
http://adc.sourceforge.net/ADC.html

[5] Facebook Sample Data Extraction
http://code.google.com/p/facebook-cpp-graph-api/.

[6] Facebook CSharp APIs
https://github.com/facebook/csharp-sdk