Specification of a P2P Protocol Built on Tor

James Lee
Computer Science and Electrical Engineering
University of Maryland Baltimore County
jlee23@umbc.edu

March 23, 2008

This document is a draft and will likely change during implementation. Expect it to be more detailed, especially with error handling, when completed.

1 Overview

The following document describes the communication between systems in a peer-to-peer (P2P) system built on Tor. This protocol will consist of simple plain text commands sent between clients and hubs. The hub will stand as a central point for directory listing, chatting, search query propagation, and results collection. A client will have the ability to connect to a single hub, chat, private message, search, display file lists, and transfer files.

All connections must be made through Tor and every client and hub must have some hidden service identified by an onion address with which they can be contacted. Clients and hubs must know this address.

2 Handshake

All connections, whether from client-to-hub or client-to-client, must begin with a handshake to prove to the receiver that the initiator is who he says he is.

A `YO <A's address> <rand1>`

B `SUP <rand1> <rand2>`

A `NOTMUCH <rand2>`

B `<COOL|NOTCOOL> [<message>]`

A opens a connection to B sending his known onion address and a random integer `rand1`. B opens a new connection to A's address sending the original `rand1` plus another random integer `rand2`. Finally A responds on the original connection associated with `rand1` sending `rand2` back to B. Now B knows that

A's address is valid and communication can continue on the first connection. The second can be disconnected.

If at any point these messages should not pass as expected, the connection should be closed. If a handshake is not successful within a minute, the connection should be closed.

3 Client-to-hub Communication

Once a client connects to a hub, and a handshake is completed successfully, the hub will know the client's address and communication can continue using the following messages.

3.1 Nick Name

A client can optionally have a nickname for their complicated onion address. It's only use is in the user interface.

Client `NICK <nick>`

Hub `<COOL|NOTCOOL> [<message>]`

The hub should note the correspondence between onion address and nick name and announce the nick name to all the clients.

3.2 Broadcast Message

Clients should be able to broadcast messages to all the other clients connected through the hub. This can result in a sort of chat.

Client `BROADCAST <message>`

Hub `<COOL|NOTCOOL> [<message>]`

3.3 Search

Clients should be able to initiate search requests through the hub.

Client `SEARCH <id> <regex>`

Hub `<COOL|NOTCOOL> [<message>]`

`id` is a unique integer specified by the client to identify a particular search query. This will be important to know what query results belong to. The `regex` regular expression must be Java compatible.

3.4 Search Results

Client `SEARCHRESULT <id> <size> <filename>`

Hub `<COOL|NOTCOOL> [<message>]`

A client should be able to return search results for a query back to the hub. For each hit, this message should be sent. `id` is the same `id` from the original query.

3.5 Disconnection

Client `DISCONNECT [<reason>]`

Hub `<COOL|NOTCOOL> [<message>]`

This message will be sent to the hub just before the user disconnects.

4 Hub-to-client Communication

4.1 Client Connection

Hub `CONNECT <address>`

This message gets sent to all logged in users when a new user logs on to the hub.

4.2 Client Nick Change

Hub `NICK <address> <nick>`

This message gets sent to all logged in users when a user sends a `NICK` message to the hub.

4.3 Broadcast Message

Hub `BROADCAST <address> <message>`

This message gets sent to all logged in users when a user sends a `BROADCAST` message to the hub.

4.4 Search Results

Hub `SEARCHRESULT <id> <address> <size> <filename>`

This message gets sent to the user belonging to a query's `id` when it receives a `SEARCHRESULT` message from another user.

4.5 Disconnection

Hub `DISCONNECT <address> [<reason>]`

This message get sent to all logged in users when a user sends a `DISCONNECT` message to the hub.

5 Client-to-client Communication

5.1 File List

A `FILELIST`

When a client receives this message, they should immediately send a file list in an XML format as yet to be documented, then close the connection.

5.2 Get File

A `GET <offset> <filename>`

When a client receives this message, they should immediately send the data contained in `filename` starting at `offset` bytes.

6 Handling Errors

Should a system receive an invalid message, they may send a `NOTCOOL` message to inform the other system of the error. They may also choose to disconnect if it seems more appropriate. Any system should be able to handle disconnection at any time.