# Building an Anonymous Community on Tor

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

March 22, 2008

**Keywords** Anonymous Communication, Tor, P2P, Hub, Community

**Description** To provide simple means to build communities of anonymous users which may talk and share files with each other through the design and implementation of a Direct Connect-like peer-to-peer (P2P) service that uses Tor for all underlying communication.

**Budget** 100 hours and $5.00

**Deliverables** Protocol documentation, source code of implementation, simple client installer, final report, presentation, and slide show.

## 1 Executive Summary

To communicate anonymously with others means that an observer is unable to identify the source, destination, and message of an intercepted packet. Tor is a software project which anonymizes arbitrary TCP communication by routing packets through multiple nodes before reaching their destination. It also provides a mechanism to hide services so that they are only accessible by a pseudonym on the Tor network. However, users must know the pseudonym in order to establish a connection to the hidden service. Additionally, hidden services are difficult for novice computer users to set up, requiring knowledge of private key management and TCP/IP addressing. Because of this and the difficulty of discovering hidden services, there are no anonymous communities on the Tor network.

I intend to make the use of Tor to communicate and share files so easy that anonymous communities can begin to grow. To do this, I will design and implement a hub-based P2P protocol similar to the popular Direct Connect service. The protocol will require that users communicate over Tor and are identified only by pseudonyms, not IP address like typical P2P protocols. The

client implementation will bundle a Tor client and configure it automatically. It will also have a basic graphical user interface (GUI) and a one-click installer.

## 2 Motivation

I am proposing this project because I want to be able to participate in a community anonymously.

Why would anyone want to remain anonymous? Users may want to remain anonymous because they have material that is illegal or for which the distribution is illegal. The material might just be embarrassing if not illegal. They may also fear retribution for communicating a sensitive or offensive message. Anonymous networks provide a great way for users to voice opinions. They also give a voice to people who are otherwise censored by a governing body.

Typical communication on the Internet is not anonymous. All users can be identified by their unique IP address. Anonymous networks such as "mixes" as originally proposed by David Chaum in 1981 route encrypted messages through multiple nodes in a network. As a result, no single node can discover the source, destination, and contents of a message.

Why would someone want to participate in a community? Communities are collections of users with some shared interest. All the users have a way to discover all the other users in the community, that way they can meet other people with a shared interest. In communities where users are identifiable, usually by a nick name, it is possible to build a reputation with the other users, even develop friendships. Simply put, communities are a fun and easy way of interacting with other people.

How can a community be built? Most importantly, there must be some way for users to know the other members of the community. Users could be known could be by nick name or pseudonym. The users must be easy to discover. If users can't be discovered, there is no way for someone to find someone else with whom to share their common interest. Most users will also need some reason to join a community. Some will expect some sort of content. A community must also be easy to join. If it is too difficult, users will not feel it is worth their time. I am proposing that the ability to communicate and share files anonymously will entice users to join an anonymous community.

Users will have those abilities through the design and implementation of a new P2P protocol similar to Direct Connect. They will be kept anonymous by Tor.

Why base the design off Direct Connect? Direct Connect is known for its simplicity. Users connect to a hub which keeps track of who is online in the community, that way new users are easily discoverable. Hubs also facilitate searching and chatting since they serve as a single broadcast point to all users. Also important is that having distinct hubs allow multiple communities to grow, each with special areas of interest. A system similar to Direct Connect will give users significant incentive to join.

Why use Tor to anonymize communication? Tor is a well-studied, mature

implementation of onion-routing. Onion-routing routes messages like Chaum's mixes, but on persistent routes, reducing latency and increasing throughput which is ideal for transferring files. Tor is the largest active anonymity network and is the subject of academic scrutiny, the benefits of which applications built on Tor receive automatically. Tor is unique in that it can route arbitrary TCP streams. It provides a standard SOCKS proxy interface so most networking libraries can use Tor unmodified. Tor also provides a mechanism to hide services so that they are only accessible by a pseudonym on the Tor network. Clients in my system will be hidden by this mechanism and users will be identified by the pseudonym that Tor provides.

# 3 Previous Work

There is a countless number of ways to build a community on the public Internet. Very few of them guarantee anonymity and none of them are built on Tor. Most of the work on Tor attempts to make it easier for a single user to join the network. For example, Torbutton is an extension to Firefox which gives users a one-click way to enable access to the Tor network. Its ease of installation and use make Tor more popular to users who wish to remain anonymous to public Internet sites. Another popular utility, Vidalia, provides a very easy GUI for managing Tor, but provides no other means for Tor users to communicate. That functionality must be provided by another program.

One such program which enables Tor users to communicate anonymously is TorChat. Clients are configured to act as hidden services on the Tor network, each one being assigned a unique 16 character pseudonym. Then a user may talk to another single user identified by their pseudonym. TorChat is not suitable for building an anonymous community. Due to its decentralized design, the only way for users to communicate with other users is to already know them and ask them for their pseudonym. This make discovering new users very difficult.

However, TorChat does make setup easy on Windows. It comes bundled with a Tor client which it configures and runs automatically. The Linux client, on the other hand, requires users to configure Tor manually, which might restrict them from joining. For a community to be built, it must be easy for as many people as possible to join, and that means considering all the major operating systems equally. My project will, like TorChat on Windows, bundle, install, configure, and run Tor automatically and transparently on both Windows and Linux so that it is easy for as many users to join as possible.

My project will build it's community around hubs based on Direct Connect. Direct Connect has proven community building qualities. For example, several Direct Connect hubs have existed on the UMBC campus for years. A student can download any Direct Connect client, type the IP address of the hub, and begin interacting with other students. Users generally discover the address of the hub through word of mouth. The pseudonym of an anonymous hub can be discovered the same way.

Once students have joined the hub, they have the ability to chat and share

files with other students. They can ask questions about classes, voice opinions about instructors, or talk about campus events.

Direct Connect is a very simple text-based protocol. The ease of its implementation means that thousands of hubs like the one at UMBC have developed around the world, each with its own community serving a special interest. Some hubs might be built around movies, other around anime.

The protocol requires that users are identified by an IP address, though. The IP address is revealed whenever a user establishes a connection with another user. It can be discovered using most operating systems' network monitoring utilities, but many clients report it in the user interface. Users of the UMBC hub quickly discovered how to associate an IP address with a residence hall, and even floor. Members of the hub who also worked in OIT could even associate IP addresses with room numbers.

I want to take the community building qualities of Direct Connect and apply the anonymity techniques employed by TorChat to create an anonymous community.

# 4 Specific Aims

There is no anonymous community based on Tor. I intend to solve this though the design and implementation of a simple Direct Connect-like P2P service. For the community to grow, the service must be easy to join and administer, installing in one click on Windows and Linux and requiring no configuration.

# 5 Plan

For the service to work on both Windows and Linux easily, all of the implementation will be done in Java, which is portable without recompilation.

The anonymity of the system will be provided by Tor. To meet the requirement of being easy to install and configure, a Java library will be developed to interface with Tor. Since Java libraries are zip files, the Tor binaries can be stored inside and self extracted to a temporary directory at runtime. The library will provide an interface to other programs to anonymize services. A program will have the ability to request a service running at a given TCP port to be hidden. The first request for a hidden service will trigger the library to start an instance of Tor. The library will make the required configuration changes to the running instance of Tor through its control protocol. Finally, the library will return the pseudonym of the hidden service back to the application as reported by Tor. Subsequent requests for hidden services will use the already running Tor process.

Applications will also need the ability to tell the library that it is done with Tor. When all applications have reported that they are done, the library should kill the running Tor process.

Next, a protocol will be defined for client-to-hub communication and client-to-client communication. The hub will stand as a central point for directory listing, chatting, and 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.

A client must have the ability to connect to a hub through Tor by its hidden service pseudonym and be able to tell the hub its own pseudonym. When the hub receives a connection, it will add the user's pseudonym to a list of online users (directory), and send the list back to the client. It must also notify the other online users that someone logged on so that they may update their own directory.

Clients must have some way to prove who they are to the hub, otherwise any user could send messages to the hub claiming to be someone else's pseudonym. So in addition to sending back a user list to the client upon connection, the hub will associate a random key to the pseudonym and tell it only to client who connected. Any subsequent messages the client sends to the hub must include both their pseudonym and their assigned key. Any other user won't know this key so they won't be able to pretend to be someone else.

A user must be able to request a disconnection from the hub. When the hub receives that message, or the connection times out, it will remove the user from the directory and notify the other online users of the disconnection so that they may update their own directory.

A user will be able to chat with other users by sending a message to the hub. When the hub receives that message, it will remove the client's key and send it to all of the online users which can handle it appropriately.

A user will be able to initiate searches of the other users' shared files by sending the query to the hub. The hub will strip out any identifying information and associate a random ID with the query so that only it will be able to identify the source. The hub will pass on the ID and query to the online users. Users can then send results back to the hub along with the original query ID, their pseudonym, and key. The hub will pass the results less keys back along to the user associated with the query ID.

Client-to-client communication presents a different challenge. How can a user prove they are who they say they are? If user with pseudonym A connects to B, but claims to be C, B must have some way to tell that A is lying. I propose using a simple three-way handshake. For example, a user connects to B and claims to be A. B will establish a new connection to A and send a random key. If B receives the same key back on the original connection, then the user on the other end is probably A. The only way it might not actually be A is if the client on the other end is colluding with the actual user A or if the user on the other end guesses the key. The key can be made sufficiently complex so that won't happen.

Alternatively, clients could generate their own public/private key pair and advertise the public key through the hub. Then all traffic from client to client could be signed. I feel this adds unnecessary complexity, but might be reconsidered in the future.

Clients will have the ability to send and receive simple text messages, file lists, and files themselves in a direct connection after a successful handshake as described above.

The protocol described above will be formally documented, then implemented in the form of two separate Java applications. Each will make use of the Tor configuration library which I described. The hub implementation must be able to run as a daemon requiring no user interaction. The client implementation will have a GUI through which a user can interact and the system can provide feedback. The GUI will be implemented using the SWT toolkit, which provides a consistent application interface to most operating systems' native GUI toolkits.

Finally, in an effort to make the client even easier to use, I will create a Java Web Start launcher for it so that it can be installed in one click on Windows and Linux.

# 6 Deliverables

The hub and client protocols will be formally documented. The Java source code of their implementation will be provided. A Java Web Start launcher for the client will be created. A final report, presentation, and slide show will be created.

# 7 Issues

The most difficult issue that I foresee is how to realistically test a P2P system. It would be impractical to actually install the client on several computers. Fortunately Linux provides means to create virtual network interfaces. Using that capability, I can bind multiple instances of the client to different virtual network interfaces.

# 8 Bibliography

*To be completed...*

# 9 Biographical Sketch

I am a senior Computer Science student at the University of Maryland Baltimore County. As an eight-year Linux user, I have developed a strong interest in free desktop software, which I contribute to frequently.

My finest application is iriverter, started in early 2005. It is a Java and SWT frontend to Mencoder with the purpose of making converting videos for portable multimedia players as easy as possible. It features a Java Web Start launcher and bundles Mencoder binaries inside itself.

I have also made significant contributions to the DD-WRT router firmware project, integrating the SixXS IPv6 configuration client into DD-WRT's web configuration interface.

I have networking experience gained from writing an HTTP client library and curses interface as a project in my networking class and a song metadata submitter compliant with the Audioscrobbler specification.

# 10 Schedule

The following deliverable items will be completed by the dates listed below.

| Date | Milestone |
|---|---|
| March 9 | Tor configuration library complete |
| March 16 | Protocols defined and documented |
| March 23 | Hub completed |
| March 24 | Progress report |
| March 30 | Client completed |
| April 6 | User interface and installer completed |
| April 14 | Complete draft report and draft presentation |
| April 21 | Final presentation |
| May 5 | Final report |

# 11 Budget

This project will take a significant amount of time. An estimation of that time is presented below.

| Item | Direct Cost | Indirect Cost | Total Cost |
|---|---|---|---|
| Project | 95 hours | 44.65 hours | 139.65 hours |
| Tor configuration library | 10 hours | 4.7 hours | 14.7 hours |
| Protocol documentation | 10 hours | 4.7 hours | 14.7 hours |
| Hub implementation | 10 hours | 4.7 hours | 14.7 hours |
| Progress report | 5 hours | 2.35 hours | 7.35 hours |
| Client implementation | 15 hours | 7.05 hours | 22.05 hours |
| User interface and installer | 15 hours | 7.05 hours | 22.05 hours |
| Presentation and report | 30 hours | 14.10 hours | 40 hours |
| Printing paper | $5.00 | $2.35 | $7.35 |

# A Research Conference

Tor was introduced at the $13^{\text{th}}$ USENIX Security Symposium.