

Forum:[Hack Forums](#)

Thread: [is it possible to create a p2p chat with public and indexed groups?](#)

Mhanzh - January 26, 2021, 2:34 am

I would like to know if it is possible create a peer to peer chat with public and indexed groups like on telegram

Thanks in advance :D

ArbiterLenny - January 27, 2021, 1:28 pm

What language are you writing it in?

Mhanzh - January 28, 2021, 1:20 am

(Jan 27, 2021 02:28 PM)Paft_ Wrote: [\[CITE\]](#) What language are you writing it in?
Java

ArbiterLenny - January 28, 2021, 7:23 am

(Jan 28, 2021 02:20 AM)Mhanzh Wrote: [\[CITE\]](#) Java

It's definitely possible, but unless you want the people on the service swapping each others IPs, I think you'll have to set up some type of relay sever to establish the connection between the clients. I'm not 100% on Java's ability to function as a backend, but Java can definitely handle the client side of things.

As far as the p2p aspect, I would consider using some type of encryption shared between the clients to encrypt the traffic between them. I'm sure there's an encryption library that could be set up on Java.

Mhanzh - January 28, 2021, 11:25 am

(Jan 28, 2021 08:23 AM)LunarLenny Wrote: [\[CITE\]](#) It's definitely possible, but unless you want the people on the service swapping each others IPs, I think you'll have to set up some type of relay sever to establish the connection between the clients. I'm not 100% on Java's ability to function as a backend, but Java can definitely handle the client side of things.

As far as the p2p aspect, I would consider using some type of encryption shared between the clients to encrypt the traffic between them. I'm sure there's an encryption library that could be set up on Java.

But is the indexing of groups possible? By installing "links" on clients to lead to groups automatically?

However, thanks a lot :D

(Jan 28, 2021 08:23 AM)LunarLenny Wrote: [\[CITE\]](#) It's definitely possible, but unless you want the people on the service swapping each others IPs, I think you'll have to set up some type of relay sever to establish the connection between the clients. I'm not 100% on Java's ability to function as a backend, but Java can definitely handle the client side of things.

As far as the p2p aspect, I would consider using some type of encryption shared between the clients to encrypt the traffic between them. I'm sure there's an encryption library that could be set up on Java.

I think to have the pgp key installed locally (each "account", of course). In this way you only have the pgp (or other asymmetric) key locally to validate that this is the real user.

Regarding the download of the data, I would have it downloaded locally through the online peers who have already downloaded it, obviously using various relays exactly as Tor does.

I'm a noob at programming, I've never done such a thing. Could you tell me if it is feasible?

ArbiterLenny - January 29, 2021, 9:28 am

(Jan 28, 2021 12:25 PM)Mhanzh Wrote: [\[CITE\]](#) But is the indexing of groups possible? By installing "links" on clients to lead to groups automatically?

However, thanks a lot :D

I think to have the pgp key installed locally (each "account", of course). In this way you only have the pgp (or other asymmetric) key locally to validate that this is the real user.

Regarding the download of the data, I would have it downloaded locally through the online peers who have already downloaded it, obviously using various relays exactly as Tor does.

I'm a noob at programming, I've never done such a thing. Could you tell me if it is feasible?

I'm sure it's feasible, but it's going to be one hell of a project.

The Tor system was heavily funded by the government. So doing something similar in regards to relays would be a headache.

What you could possibly do is to use an RSA algo to verify users by sending a public key to the network and having multiple users verify it. I'm kind of speaking out of my ass, but I'd look into RAS and AES encryption.

Also, I wouldn't use a single stored local key, I would make it dynamic so that it couldn't be hijacked.

Mhanzh - January 29, 2021, 12:57 pm

(Jan 29, 2021 10:28 AM)LunarLenny Wrote: [\[CITE\]](#) I'm sure it's feasible, but it's going to be one hell of a project.

The Tor system was heavily funded by the government. So doing something similar in regards to relays would be a headache.

What you could possibly do is to use an RSA algo to verify users by sending a public key to the network and having multiple users verify it. I'm kind of speaking out of my ass, but I'd look into RAS and AES encryption.

Also, I wouldn't use a single stored local key, I would make it dynamic so that it couldn't be hijacked.

Come to think of it, yes, relays are quite difficult. In fact, to have an anonymous chat it would be enough to implement the possibility of using Tor. In any case, thank you very much for the answers! : D In case you have any questions (again, I'm noob, can I contact you via convo?)

ArbiterLenny - January 30, 2021, 5:34 am

(Jan 29, 2021 01:57 PM)Mhanzh Wrote: [\[CITE\]](#) Come to think of it, yes, relays are quite difficult. In fact, to have an anonymous chat it would be enough to implement the possibility of using Tor. In any

case, thank you very much for the answers! : D In case you have any questions (again, I'm noob, can I contact you via convo?)

Of course! I'd be happy to help.

GenericSkid - January 31, 2021, 4:02 pm

(Jan 28, 2021 12:25 PM)Mhanzh Wrote: [\[CITE\]](#) But is the indexing of groups possible? By installing "links" on clients to lead to groups automatically?

To identify a user you can randomly generate a UUID when they first open the app. This will be their permanent ID. The display name can be changed/different but all internal references will use this.

How you implement the P2P depends on how you want the chat experience to function. Lets cover how to discover users first:

Case 1: All on the same network

If you're all on the same network you can use something like JGroups. There's even a [chat-system tutorial for using it for this purpose](#). You'll eventually want to make things a bit more advanced than the tutorial code, for example custom message types for different chat actions like chat-messages, media-messages, username-changes, etc.

The important important thing about JGroups is that its just a one-liner to join a group and all other users of the group are discovered automatically.

The problem though is that if you are not on the same network, then you can't communicate with others.

Case 2: All on different networks

If you're all on different networks you can't find other users like you would on a LAN. Some ways to get around this are to have a couple of servers that users will ping. If a user stops pingging the server then the server assumes the user has disconnected. A user can also request who is online from the server. You can write an efficient server to do all of this using Netty.

Once you have a group of users you can communicate with them by: Direct: Initiate a socket connection to those peers

Indirect: Send a packet to the server with a list of peers that are intended to receive the message, and the server will pass it along

If you go the direct route, your local client will have to have lots of open sockets to communicate with the peers, and each client will know the connection information of the peers.

For the indirect route, all the logic belongs on the server. And the users are only told whatever the server tells them.

For managing groups with the LAN approach, you'll need a way to discover those groups. Running a dedicated channel that allows users to publish new groups they've made could work.

For the multi-network approach, you can have the server track what groups users have made/deleted/etc and then broadcast that to new clients when they connect.

Recommendations for libraries

[Netty](#): For the server if you choose to go beyond the LAN.

[JChannels](#): For local network discovery and messaging.

For more recs I'd have to know what direction you'd want to take with the app.

Shadow Cloud - January 31, 2021, 9:25 pm

Would you mind explaining why you have the requirement for the chat to be peer to peer? There are quite a few options here depending on what you are trying to achieve.

The easiest approach would be to have peer to peer chats communicate directly (assuming this is the requirement) and having the group chats operate via a central point rather than being distributed among clients.

Personally I'd probably opt for a mix here. Have an API that deals with auth token that has to auto refresh every x seconds. If there is no refresh of the auth token, you can drop the client from the group (ie offline) by doing this you can broadcast events as they take place to anyone who is subscribed to them while retaining the ability to have a client push messages straight to another client without routing it via the server, the server would purely be responsible for maintaining and broadcasting the list of clients available to push to? If you want offline capabilities (so someone who isn't online can read the messages when they log back on, you'll need to rethink the position here, hence my question on what you are actually trying to achieve)

My suggestion would be to start with a normal chat and incorporate a "private" chat as an add-on later.

Mhanzh - February 1, 2021, 1:51 am

(Jan 31, 2021 05:02 PM)GenericSkid Wrote: [\[CITE\]](#) To identify a user you can randomly generate a UUID when they first open the app. This will be their permanent ID. The display name can be changed/different but all internal references will use this.

How you implement the P2P depends on how you want the chat experience to function. Lets cover how to discover users first:

Case 1: All on the same network

If you're all on the same network you can use something like JGroups. There's even a [chat-system tutorial for using it for this purpose](#). You'll eventually want to make things a bit more advanced than the tutorial code, for example custom message types for different chat actions like chat-messages, media-messages, username-changes, etc.

The important thing about JGroups is that its just a one-liner to join a group and all other users of the group are discovered automatically.

The problem though is that if you are not on the same network, then you can't communicate with others.

Case 2: All on different networks

If you're all on different networks you can't find other users like you would on a LAN. Some ways to get around this are to have a couple of servers that users will ping. If a user stops pingging the server then the server assumes the user has disconnected. A user can also request who is online

from the server. You can write an efficient server to do all of this using Netty.

Once you have a group of users you can communicate with them by: Direct: Initiate a socket connection to those peers

Indirect: Send a packet to the server with a list of peers that are intended to receive the message, and the server will pass it along

If you go the direct route, your local client will have to have lots of open sockets to communicate with the peers, and each client will know the connection information of the peers.

For the indirect route, all the logic belongs on the server. And the users are only told whatever the server tells them.

For managing groups with the LAN approach, you'll need a way to discover those groups. Running a dedicated channel that allows users to publish new groups they've made could work.

For the multi-network approach, you can have the server track what groups users have made/deleted/etc and then broadcast that to new clients when they connect.

Recommendations for libraries

[Netty](#): For the server if you choose to go beyond the LAN.

[JChannels](#): For local network discovery and messaging.

For more recs I'd have to know what direction you'd want to take with the app.

The second option, just that instead of letting the server do it, is it possible?

(Jan 31, 2021 10:25 PM)Shadow_Cloud Wrote: [\[CITE\]](#) Would you mind explaining why you have the requirement for the chat to be peer to peer? There are quite a few options here depending on what you are trying to achieve.

The easiest approach would be to have peer to peer chats communicate directly (assuming this is the requirement) and having the group chats operate via a central point rather than being distributed among clients.

Personally I'd probably opt for a mix here. Have an API that deals with auth token that has to auto refresh every x seconds. If there is no refresh of the auth token, you can drop the client from the group (ie offline) by doing this you can broadcast events as they take place to anyone who is subscribed to them while retaining the ability to have a client push messages straight to another client without routing it via the server, the server would purely be responsible for maintaining and broadcasting the list of clients available to push to? If you want offline capabilities (so someone who isn't online can read the messages when they log back on, you'll need to rethink the position here, hence my question on what you are actually trying to achieve)

My suggestion would be to start with a normal chat and incorporate a "private" chat as an add-on later.

Mine was mostly curiosity