Lab 1: Information flow control

Yongzhe Xu
Junjie Shan

In this lab, we are supposed to develop a dating server using Troupe. The client sends his/her profile to the server and decides who he/she wants to date.

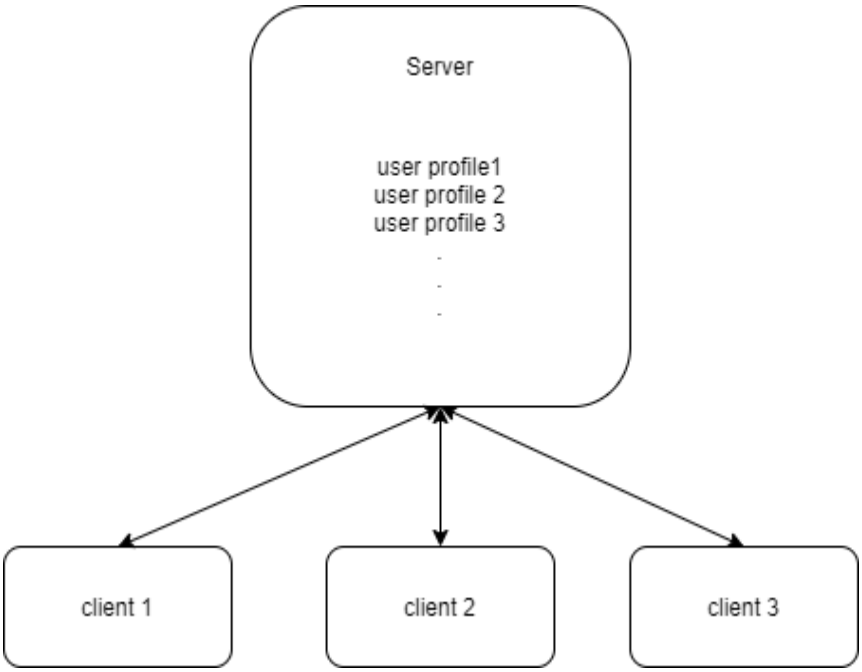
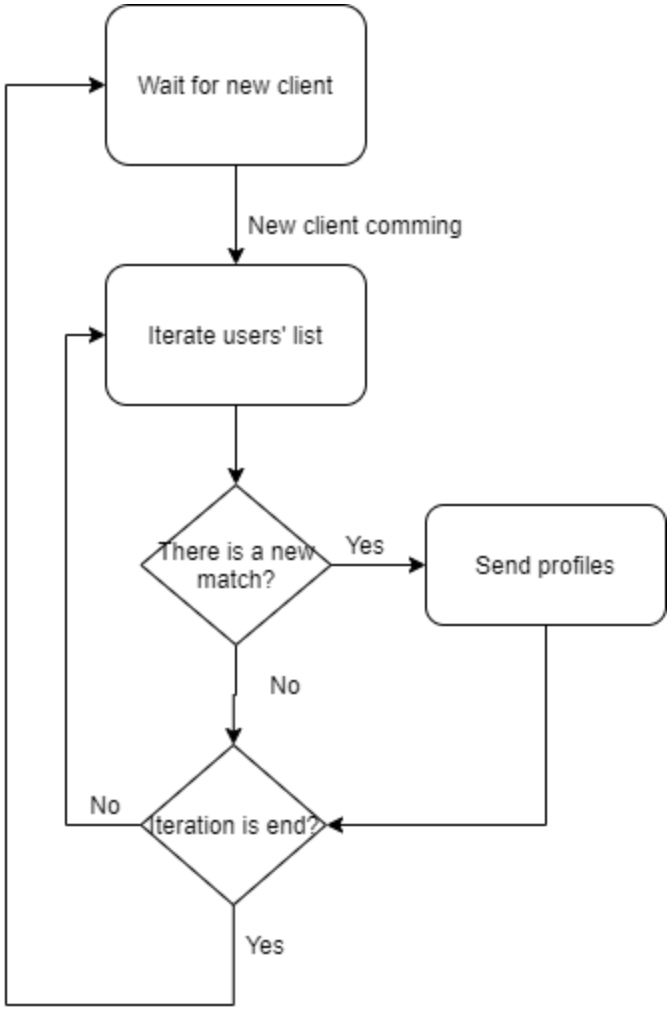


Figure above illustrates the topology of the system. There are one central server and several distributed clients. The server and client connect each other via P2P network. All clients locate the server using the fixed id of the server.



The server will store the previous clients and wait for a new client. When a new client is coming, the server will iterate the previous clients' list and try to find a match between the new client and previous client. If there is a match, the server will exchange the profiles of matched clients. If the iteration is finished, the server will wait for a new client.

Information Flow

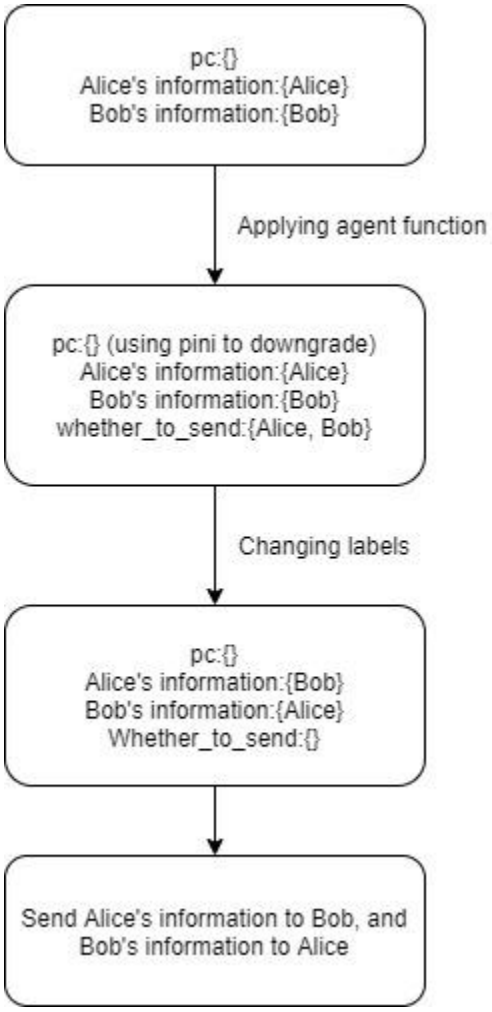


Figure above illustrates the design of the server regarding information flow. Imagine we want to compare two clients, Alice and Bob. Alice's personal information is labeled with the tag “Alice” while Bob’s personal information is labeled with the tag “Bob”.

While the server executes the agent function of both clients, the pc will rise to {the other client’s tag}. For example, if the server executes the agent function from Bob, the pc will rise to {Alice}, because Bob’s agent function will use the personal information of Alice. In order to downgrade the pc, pini is used.

After the matching is finished, we have a new variable, which indicates whether to send the profiles to each other. The variable has the label {Alice, Bob}, because it’s derived from preference of Alice and Bob. So we need to declassify the “Whether_to_send” variable. Besides, Alice’s personal information should have the label of Bob and Bob’s personal information should have the label of Alice. Otherwise their information can not be sent to each other.

Malicious client

We implemented four malicious clients in total.

Malicious client 1: To leak the number of users in the dating server. We added “send(“NEW”, lev)” in the agent function, so when the dating server tries to match our malicious client and another client, it will send a message “NEW” with the security label of another user to our malicious user, so that we can get the total number of current users in the dating server.

Malicious user 2: This is the advanced version of malicious user 1, which can leak other users’ genders. Since we denote the gender using “true” and “false”, and we cannot directly send back the gender of another user because of the security label, so we can use if(gender) then sleep 3000 else () to make the dating server sleep for 3 seconds if another user’s gender is “true”. Then, we calculate the time between two received “NEW” messages, so that the malicious client can tell leaked users’ genders.

Malicious user 3: In this malicious client, we try to find out if a specific user is currently on the dating server. In the matching function, we add “if(name = “xx”) then sleep 3000 else ()” to make the server sleep for 3 seconds if there is a user named “xx”. Then, we calculate the time between two received “NEW” messages so that the malicious client can tell if there is a user called “xx”. Also, in this way we can leak a certain user’s name, age, or interests by adding something like “if (name=”xx” and age > 35) then sleep 3000 else ()”.

Malicious user 4: Brute force attack. We create multiple user profiles using different ages, genders, and interests to match as many users as possible, and the malicious client can send all profiles to the dating server, which will try to match multiple user profiles from our malicious client with other users’ profiles and send all matched profiles back to our client. So, we can leak other users’ profiles.

Some measure to protect the server from malicious attacks

1. Add authority to the send() function. Troupe does not provide authority or secret for the send function. So every client can send messages to whoever he wants, as long as the information flow is secure. So one solution is the modify the send function to send(pid, tuple, secret)
2. Second measure is to limit the maximum number of profiles of each user. So the client will not be able to launch brute force attacks.

Contribution

Mutual efforts towards the server.

Junjie: Malicious client a,b.

Yongzhe: Malicious client c,d. Benile client.