Hybrid P2P Online Chatting App

CIS533 Liyuan Liu
Jerry
What’s Hybrid Peer-to-Peer?

- Server
- Client Jerry
- Client Carl
Motivation and Function (providing 17 commands for user, 18 functions, for MAC OS version):

1, Privacy:
   /leave
   /unleave
   /black Carl
   /unblack Carl
   /hide
   /unhide
   /busy
   /unbusy

2, Safety: RSA encryption (compare with QQ)
   /lock
   /unlock
   /update

3, Stability: Its structure.

Some basic functions:
   /register
   /login
   /logout
   /send Carl
   /close Carl
   /check

Problems: It can
1, protect user’s msg and privacy
2, reduce server’s burden
3, guarantee user’s conversation won’t be interrupted by server
Basic Function
on client side

./server
./client /register
or ./client /login

input username
input passcode twice
connect to server and recv server’s public key
generate random ticket ,RSA key,random complex string
complex and encrypt them with server’s public key
send to server and wait for reply(set up listen)

random request type len

head real_pass len of complex len of real pass pass name(Jerry) Jerry’s public key encrypted ticket
On server side

1. accept a connect from Jerry
2. send Jerry server’s public key
3. recv Jerry’s msg(encrypted)
4. decrypt msg, check it (whether pass is right, whether user name has existed, IP is right) and store it, update it (IP, Public Key, status)
5. give feedback to client (success or failure)

IP, user name, passcode, public key, ticket, status(4), IP_isolate(2), socket_fd

status -> online, IP_isolate -> UNLOCK
command send to server

/logout
/leave
/unleave
/hide
/unhide
/busy
/unbusy
/lock
/unlock

input command
if it's /update, generate new RSA key
send to server (or other clients) and wait for the reply
recv msg and show user the corresponding msg

on client side

/head   user name   another user name   new public key (NULL, unless it is /update)

/random request type len

/check
/send Carl
/update
on server side

recv msg from Jerry

check req type and do corresponding action

give feedback to Jerry

random req_type len type(success or failure)

reply with this msg

/head user 1, user 2, user 3, ...

the num of users

send Carl -> reply with Carl's ticket and IP and public key (encrypted with Jerry's Key)

head Carl's name Carl's IP Carl's Public key Carl's ticket
Command send to another client (Carl)

- `/update`
- `/close Carl`
- `/black Carl`
- `/logout`
- `head`  `new public key`
- `random`  `request type`  `len`  `my user name (Jerry)`
- `/unblack Carl`

Input command:
- generate new RSA key
- send to another client

Flow:
1. Input command
2. Generate new RSA key
3. Send to another client
Command send to another client (Carl)

/send carl

input command

send to server and wait for reply

recv msg and decrypt it

use carl’s public to encrypt auth info

send auth info to Carl and wait for reply

if Carl agrees with your connect request, store Carl’s info

start to talk (check Carl’s status)

send authentication information to Carl

head | Jerry’s public key | my user name(Jerry) | Carl’s ticket(encrypted by Carl’s public key)

send Carl the talking msg

random | request_type | len | my user name(Jerry) | talking msg(encrypted)
On Carl’s side

1. recv connect request from Jerry
2. decrypt msg, check Jerry’s ticket
3. check whether Jerry in Carl’s black list
4. send feedback to Jerry

Feedback
- random
- request type
- len
- Carl’s name
- Carl’s public key
- type

/send Jerry

input command

check Jerry’s status

if connect, Carl can talk to Jerry now
Motivation: 1. protect user's msg and privacy 2. reduce server's burden 3. guarantee user's conversation won't be interrupted by server.


Some basic functions: /register /login /logout /send Carl /close Carl /check

User send to server protocol: /login /register /lock /unlock /logout /check /send /update

**Random** req type | len | flag | len of complex str | len of pass | pass | username | public key | random | ticket

**Random** req type | len | user name | another user name | NULL unless | send | new pub key | NULL unless | update

Server send to user protocol:

Feedback: success or fail (/login /register /lock /unlock /logout /leave /unleave /busy /unbusy /update /hide /unhide)

**Random** req type | len | success | failure | which kind of success | failure

User list (/check)

**Random** req type | len | num of users online | user1, user2, user3, ... 

Send user's info to another user

**Random** req type | len | user's name (encrypted) | user's IP (encrypted) | user's ticket (encrypted)

Send client server's public key

**Random** req type | len | server's public key

User send to user (auth request, say msg, public key, close conversation request)

**Random** req type | len | sender's public key | sender's name | receiver's ticket (encrypted)

**Random** req type | len | sender's name | say msg (encrypted)

Server's Behavior

Client Behavior

!!!