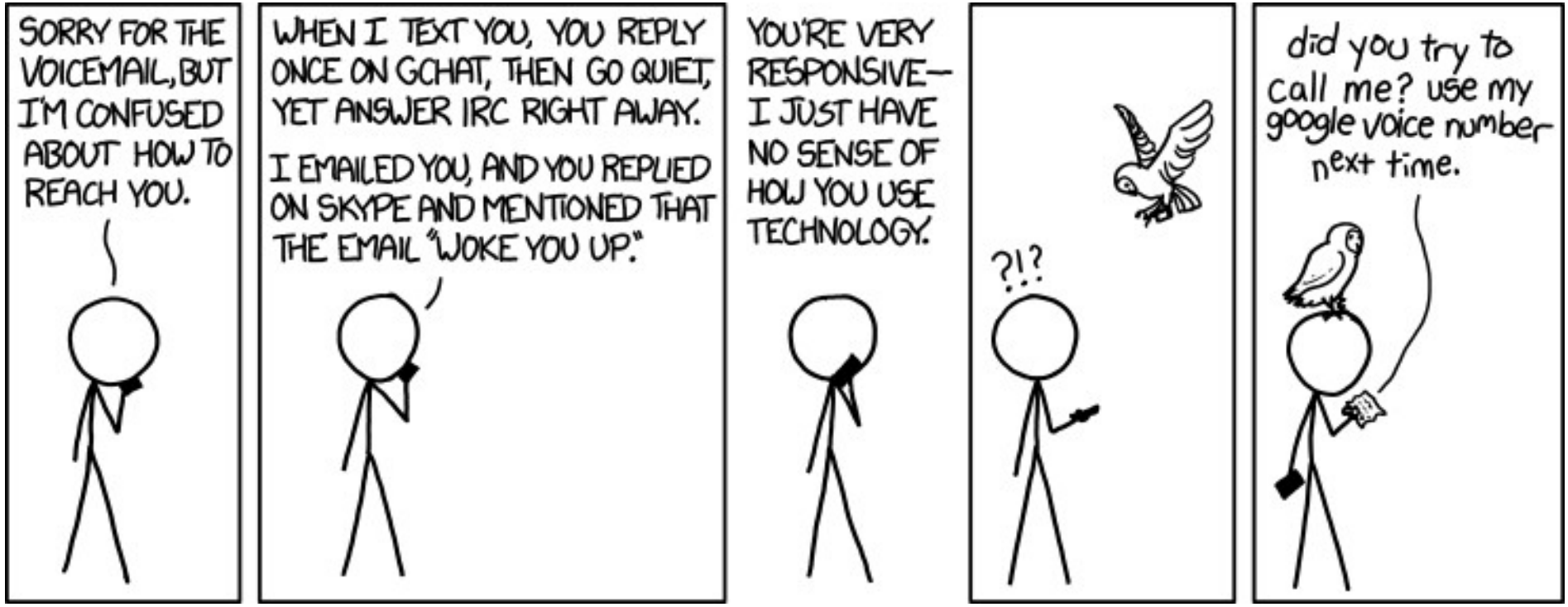

A simple chatroom server

Project goals

- **This project provides a server implementation for a chat application**
  - The server supports a simple, text-based API
  - The project also includes a simple test client
  - Depending on the assignment, the server may be running somewhere
  - Normally, for a class project, you are *not allowed to modify the server*
- **The challenge is to create clients that communicate with the server**
  - Provide an attractive user-interface
  - Support some or all of the server features
  - Provide robust error handling
- **This is open source software (BSD 3-Clause License)**
  - Feel free to take the code and use it for your own purposes
  - See the license file for further details

Overview of server features

- **The server provides the following general functionality**
  - Create/delete accounts, login/logout, change password
  - Create, join and leave chatrooms
    - *Public chatrooms: anyone can join*
    - *Private chatrooms: only the owner (creator) can add users*
  - Send messages to a chatroom or to individual users
  - Check the online status of users, list the members of chatrooms

- **If the server is left running, data is regularly cleaned up**
  - Clients are disconnected after a period of inactivity
    - *Short if not logged in (a few minutes), longer if logged in (an hour)*
  - Chatrooms and accounts are removed if not used (a few days)

Normal sockets vs. secure (SSL) sockets

- **The server and the test client support both**
  - ServerSocket and Socket – unencrypted data transfer
  - SSLServerSocket and SSLSocket – encrypted data transfer
- **You do not need to use SSL !!**
  - Both server and test-client ask if you want “secure sockets”
  - If not, just answer “no”
- **If you want to experiment with SSL, read the “Howto_SSL” document**
  - See the references at the end of the “howto” for further information

Other notes about security

- **Reminder: If you do not select “secure sockets”, there is no encryption**
  - In this case, all messages are *plain-text,* including login passwords
- **After login, the server uses token-based authentication**
  - The server sends the client a “token”
  - The client must include this token with every request
  - This provides security without requiring username/password every time
- **The server maintains no history of logins or messages**
  - The only data saved: currently defined accounts and chatrooms
  - Passwords (in the account information) are salted and hashed securely

Messaging protocol

- **The messaging protocol is plain-text, similar to the HTTP protocol**
  - Each command is a single line of text
  - Names & passwords must be at least 3 characters
  - Usernames and chatroom names must be different
  - Messages may be 1-1024 characters

- **Individual messages follow the format**
  - `MessageType [Token] Data`
    - ***`MessageType`*** *identifies the kind of message*
    - ***`Token`*** *must be sent with almost all messages from client to server*
      - Everything except Ping, Login and CreateLogin
    - ***`Data`*** *varies by MessageType*
  - Message parts are separated by vertical bars '|'

- **A minimal client only needs to implement a few message types**

Messages (client → server)

MessageType	Data	Notes
CreateLogin	Username, Password	Fails if name already taken (user or chatroom), or invalid After creating an account, you still have to login
Login	Username, Password	Fails if name/password do not match
ChangePassword	New password	Fails only if token is invalid
DeleteLogin	--	Fails only if token is invalid; after delete, token becomes invalid
Logout	--	Never fails; token becomes invalid
CreateChatroom	Name, isPublic	Fails if name already taken (user or chatroom), or invalid After creating a chatroom, you still have to join
JoinChatroom	Chatroom, User	User can add themselves to public chatrooms Only the creator can add user to a private chatroom
LeaveChatroom	Chatroom, User	You can always remove yourself Chatroom creator can remove anyone
DeleteChatroom	Chatroom	Only the creator can delete a chatroom
ListChatrooms	--	Returns a list of all public chatrooms
Ping	[Token]	Without a token: always succeeds With token: succeeds only if token is valid
SendMessage	Target, Message	Send message to user or chatroom Fails if user not online / Fails if not a member of the chatroom
UserOnline	User	Succeeds if the user is currently logged in
ListChatroomUsers	Chatroom	Returns a list of all users in the given chatroom You must be a member of this chatroom

Message types (server → client)

- **Responses to client commands**

MessageType	Data	Notes
Result	Boolean Boolean Token Boolean List	True if the command succeeded, otherwise false After a successful login, also returns the authentication token When a list is requested, also returns list results
MessageError	Error message	Incorrect commands, wrong number of arguments, etc.

- **Server initiated messages**

MessageType	Data	Notes
MessageText	Name, Target, Text	Name of user sending message Target is where the message was sent (chatroom or user) Text of the message

Sample exchange of messages with the TestClient

```
brad@fhnw-xubuntu-2018$ java chatroom.testClient.TestClient
Enter a valid IP address
127.0.0.1
Enter a valid port number
31415
Enter commands to server or ctrl-D to quit
CreateLogin|brad|mypassword
Received: Result|true
Login|brad|mypassword
Received: Result|true|4FA4563A5C2FFD1E703B49190DC348BD
CreateChatroom|4FA4563A5C2FFD1E703B49190DC348BD|CatChat|true
Received: Result|true
JoinChatroom|4FA4563A5C2FFD1E703B49190DC348BD|CatChat|brad
Received: Result|true
SendMessage|4FA4563A5C2FFD1E703B49190DC348BD|CatChat|Hello, all cat people!
Received: MessageText|brad|CatChat|Hello, all cat people!
Received: Result|true
SomeInvalidCommand
Received: MessageError|Invalid command
SendMessage|Wrong|Parameters
Received: MessageError|Invalid command
Logout
Received: Result|true
brad@fhnw-xubuntu-2018$
```

Connecting the client
→ User inputs are green

Then client messages entered by user
And server responses

Ctrl-D entered, to stop client