Presentation Projects: Introduction

Having mastered a great deal of algorithmic and programming skills, the goal of the presentation project is to challenge you further. You will be asked to work in a team of two, choosing one of the following topics. All of them can be implemented on top of your completed distributed chat system:

- Secure messaging: guard against eavesdropping
- Reliable messaging: protection against a bad communication channel
- Online gaming: extend the chat system with game functionality
- Parallel search engine: turn the chat system into a distributed search engine

Groups can propose different projects, but they must (1) be approved with instructors' consensus, (2) have an equivalent degree of difficulty, and (3) extend the chat system (ideally).

You will be asked to make a presentation with a live demo in front of the whole class during W14.

Secure Messaging

Turing and his friends built one of the first computers to crack the Enigma code. Every technology, by itself, stands as a double-edged sword. As a user, you have the right to protect your transactions against eavesdropping.

But the question is: *how*?

The intuitive idea is, of course, to scramble your message, a process that we call encryption. You want your friend to read the clear message, so you will need decryption as well. This takes for the form of an agreement between you and your friend before you send your scrambled message.

On the Internet, everything said is in the open. So there is no hideaway in which you and your friend can meet. So how is it possible to establish such agreement, and then communicate?

The field of computer security (and a related field, data privacy) is a vast and interesting research area. In this project, however, we only want to get a taste of it.

The goal: To get two chat clients to communicate to each other securely, meaning that the server who is passing the messages in between has no idea what they are talking about. You

will need to implement a shared key protocol as well as a simple encryption/decryption algorithm for the client side, and make this work!

Reference: MacCormick, 9 algorithms, **Chapter 4**.

Reliable Messaging

Going over your homework with your classmate in a noisy location is challenging. It has a lot more to do with the noise than what you drink.

Interestingly, this is one area in which a computer seems to do slightly better than a human. The idea, as covered in the 9-algorithm book, is to tell the other party just a little bit extra. These extra bits carry enough information to recover what the environment might have corrupted from your message.

If you are interested, you should read more on information theory, a discipline which giants such as Claude Shannon helped build. But here, we just want to get a taste of it.

In the chat system, the server is responsible for passing a message from A to B (two chat clients). **The goal: Your server should emulate the "noisy location" by flipping bits of the messages randomly. Your next task is to implement a checksum on the client side, so that despite a bad server, the clients read each other's' messages without a problem.**

You could try this algorithm with your friend in the noisy bar, and start to appreciate why computers do better than you!

Reference: MacCormick, 9 algorithms, Chapter 5.

Online Gaming

Online gaming is one of the fastest growing industries in the world today. It is a huge, interdisciplinary field which brings together artists and creative professionals (asset generation and narrative development), business executives (marketing, logistics, distribution), and of course talented computer science professionals (game engine design, system infrastructure and implementation, QA). This project topic is designed to give you a taste of some of the complexities behind implementing a simple game. The game shall use the chat system as its backbone.

In the chat system, the model of communication for chatting across clients is to send a message to the server, have the server forward the message to a client (or set of clients), and then have the server relay responses. Essentially, our server is our “man in the middle” which the clients rely upon for sending and receiving messages.

The goal: To adapt this model for implementing a simple game (such as Tic-Tac-Toe, which is the suggested game to implement). Two clients playing a game can send their moves to the server, which will evaluate them, maintain a consistent game state, and relay the changes in game state to the clients as appropriate. Incidentally, this model is conceptually similar to how many popular and successful online games actually work.

Parallel Search Engine

Have you wondered how many search requests Google gets? The latest report is 100 billion monthly. This translates to around 40,000 hits per second. This will melt down one single machine.

What any big search engine does is to partition its web page index among *N* machines, have each of them process a query, and then merge (reduce) their responses.

The goal: To use one client to send a search query on some index to the server, and then have the server forward partitioned requests to *N* other available clients. Each available client indexes a portion of the total text. Their responses should then be merged by server and “the answer” returned to the querying client.

The bottom line is that you need to make distributed search architecture work. Another challenge (out of the scope of this project) is to plan what happens when the corpus is so big that it does not fit in the memory of your machine: things can get considerably more complex.