Tic Tac Toe game with Superchat
(Assignment 1)

Advanced Multimedia Technologies

Author	Ladislav Šulák
Email	er1281@edu.teicrete.gr
Institute	Technical Educational Institute of Crete

Heraklion 11/2017

Objectives

The aim of this project was to implement a web application in which there will be two users communicating with each other via chat as well as video and audio. Additionally, there will be a tic tac toe game implemented with the use of Canvas object. Each time a user makes a move his page sends a message to the other user through the websocket or webrtc data channel. In this message there is information about the user and the move itself. Then the page of the other client makes the same move in order to keep tic tac toe terrain updated.

In the section Implementation there will be described the implementation and some technical details as well as software requirements and in the section Communication protocol there will be communication protocol described. The last section Conclusions will briefly summarize the project and propose some possible future improvements.

Implementation

During the implementation phase there were used typical languages for web development like HTML, CSS and Javascript. The background which was most interesting part was written in Javascript with socket.io[1] and Node.js[2] libraries and for tic tac toe visualization it was used canvas object with its operation performed also in a form of Javascript functions. The project has not object oriented paradigm, it consist mostly of a set of functions in:

- **app.js** - it implements a server part of our web application which listens on a specific IP address and port. It handles client messages (see the next section) and it uses secured communication via self signed certificate and 4096 bit RSA key generated from OpenSSL library[3]. It utilizes Node.js library for which it can be said that it is asynchronous, event-driven framework written in Javascript which has bunch of C/C++ code under the hood for operations connected with IO, operating system and so on. For our purposes we used port 443 and IP address locally allocated by our router, but those can be changed very easily to another ones.

- **index.html** - it implements a client part of the application and this code is emitted to client web browser once he or she connects to IP address and port specified in **app.js**. A design and the whole frontend as well as game functionality in the canvas object is implemented in this file. Regarding the game, it was inspirated and partially taken from[4]. The game will be reset on all clients if it is finished or submitting the reset button. The game board has 10x10 fields and for the win it is needed to obtain 4 consecutive fields, but these constants can be changed in the code very easily as well as canvas and section sizes, which are also constants. Only the client remembers a current state of the board which means that if a new client will connect, he has no information about previous moves and state of the game. This was done because in the requirements of the project it was needed to have this application between 2 clients and after all, the game can be player only with 2 players at the same time. However, if one player disconnects and the other client will come or he is already connected and waiting, the game will be restarted and such player can play. Additionally, all message handling in the client part of our application is done here. Regarding communication protocol, it will be detailed in section Communication protocol. There is also chat in which multiple users can participate in and it was taken mostly

[1] https://socket.io/
[2] https://nodejs.org/
[3] https://www.openssl.org/
[4] https://codepen.io/solartic/pen/qEGqNL

from the lecture materials of the course and the code is also available online[5]. Regarding audio and video stream transfer and handling, it was completely taken from materials from the course. There were done just some minor details and improvements like responsive design which is very useful if a client want to use mobile or other devices with different resolutions.

- **package.json** - settings in JSON format necessary for Node.js part of the application.

Besides these three files there are also following in the project itself:

- **beep.mp3** - a short sound which is played if the used successfully clicks on the field in the game and makes the move.
- **bg.jpg** - the picture which is being used as a background in the frontend.
- **favicon.ico** - icon which can be seen in the browser once the client will connect and loads page content.
- **jquery-1.10.1.min.js** - jQuery library[6] downloaded locally.
- **cert.pem, key.pem, csr.pem** - contain public RSA key and self-signed certificate used for making a secure connection.

The application was tested on Fedora 26 operating system with 64-bit architecture with Google Chrome web browser with version 62. It was tested also on Android 7.0 operating system in Huawei P9Lite phone with Google Chrome web browser with the same version. During testing phase it was all done in local network in space allocated by router, so it was not available from the Internet and public space. However for making it online, the following steps were done:

1. Registration of the domain on NoIP service[7], which is basically needed for *domain name* $\leftrightarrow$ *IP address* translation with the usage of dynamic DNS service.
2. Downloading and setting up the application from NoIP service[8] which will send a public IP address of the device where the server part of the application runs to NoIP service. Such record is saved so we don't have to deal with IP change in NoIP settings on our network change. This check works as a daemon which runs in the background of the operating system.
3. Port forwarding must be set in the router configuration. It forwards the traffic from the router on port 443 to the machine in the local network. This ensures that once the client from the Internet will connect to server part of the application to the domain registered in NoIP, so to our public IP address on port 443, the router will forward this connection to the device. For this action it was needed to open port 443 which required to change the firewall settings in iptables[9] as well. For checking the availability of the port over the Internet it was used Port Check and IP detection tool[10].

On the pictures 1 and 2 are screenshots of the application. Both were taken from a different game round and both are on different devices (notebook vs smartphone).

[5] https://openclassrooms.com/courses/ultra-fast-applications-using-node-js/practical-exercise-the-super-chat
[6] https://jquery.com/
[7] https://www.noip.com
[8] https://www.noip.com/download
[9] http://ipset.netfilter.org/iptables.man.html
[10] www.portchecktool.com

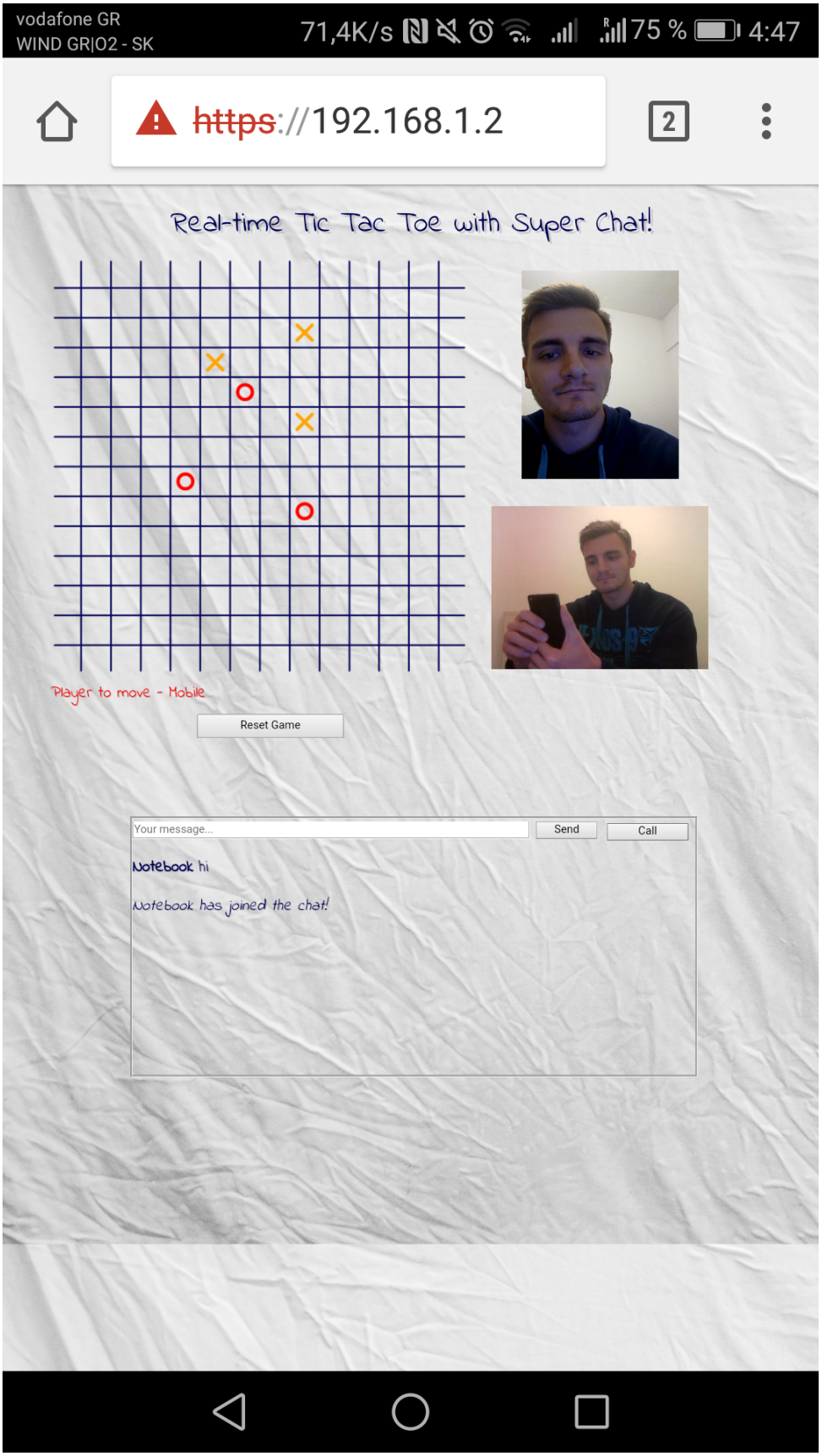

Figure 1: Screenshot of gameboard played on mobile phone Huawei P9Lite with notebook. Game was being played and the winner needs 4 fields to achieve. Audio and video stream were also working and the size of video box was adjusted. Connection was considered to Chrome as 'Not secure' because the certificate was not signed by any of known Certificate Authorities.

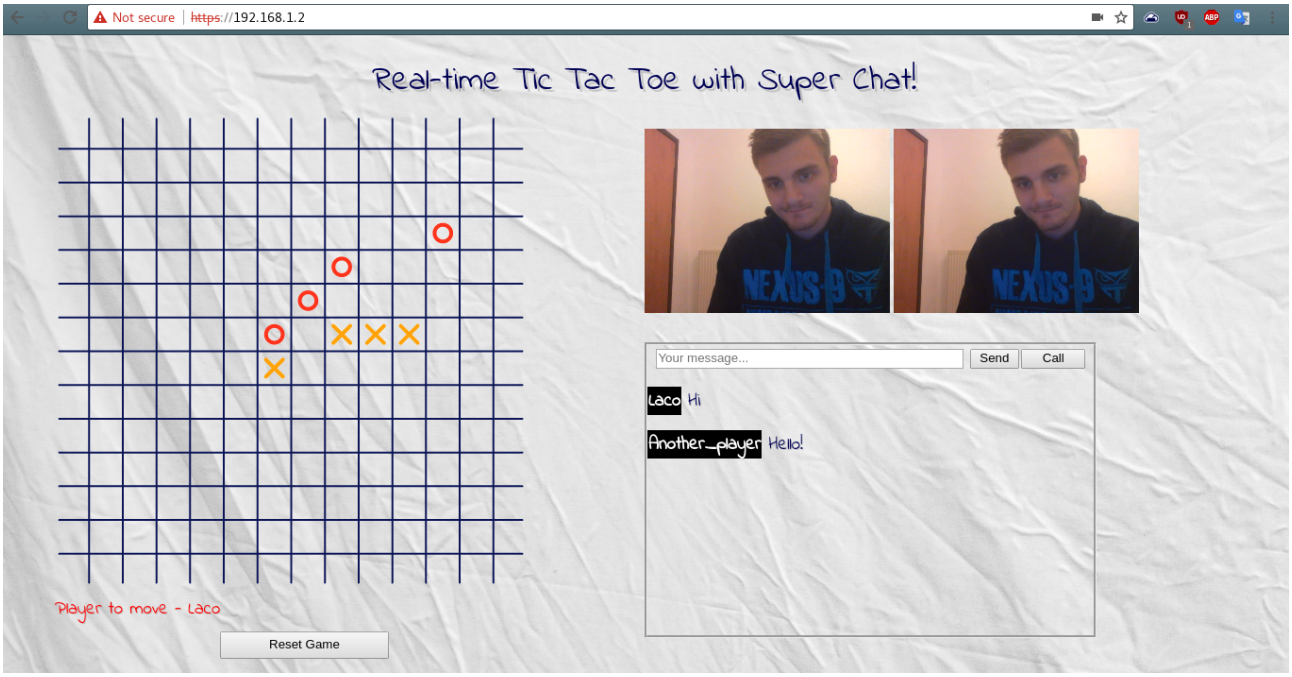

Figure 2: Screenshot of gameboard played with notebook on 2 tabs in Chrome web browser. It is similar than in 1.

Communication protocol

The communication protocol is depicted in the sequence diagrams: 3 and 4. It's not just about communication between server and client or client and client but also about the logic of the game and the application itself.

Conclusions

All required objectives in this project were successfully accomplished. It should be noted that there can be more clients connected to NodeJS server and they all can participate in chat communication. Regarding audio, video and tic tac toe game, it is always just between two users, but when some client disconnects, the other one can replace his place in case of more active users which are waiting for the game and audio and video stream. Additionally, the application is secured via self-signed certificate which uses RSA key for data encryption. The design of the application is responsive, which means that it can be used in devices with different resolutions. In fact, application was also tested on mobile phones as well as notebooks and it was working in all cases.

Regarding possible future improvements there is place for better user improvements (like actual score) or setting up the game in its very beginning (number of fields in the game board, number of fields to win) which is done only in the code and therefore for users more difficult to change. Also it would be nice so that the order of players will switch on win or restart, for example for winner of the round will have a first move in the next one.

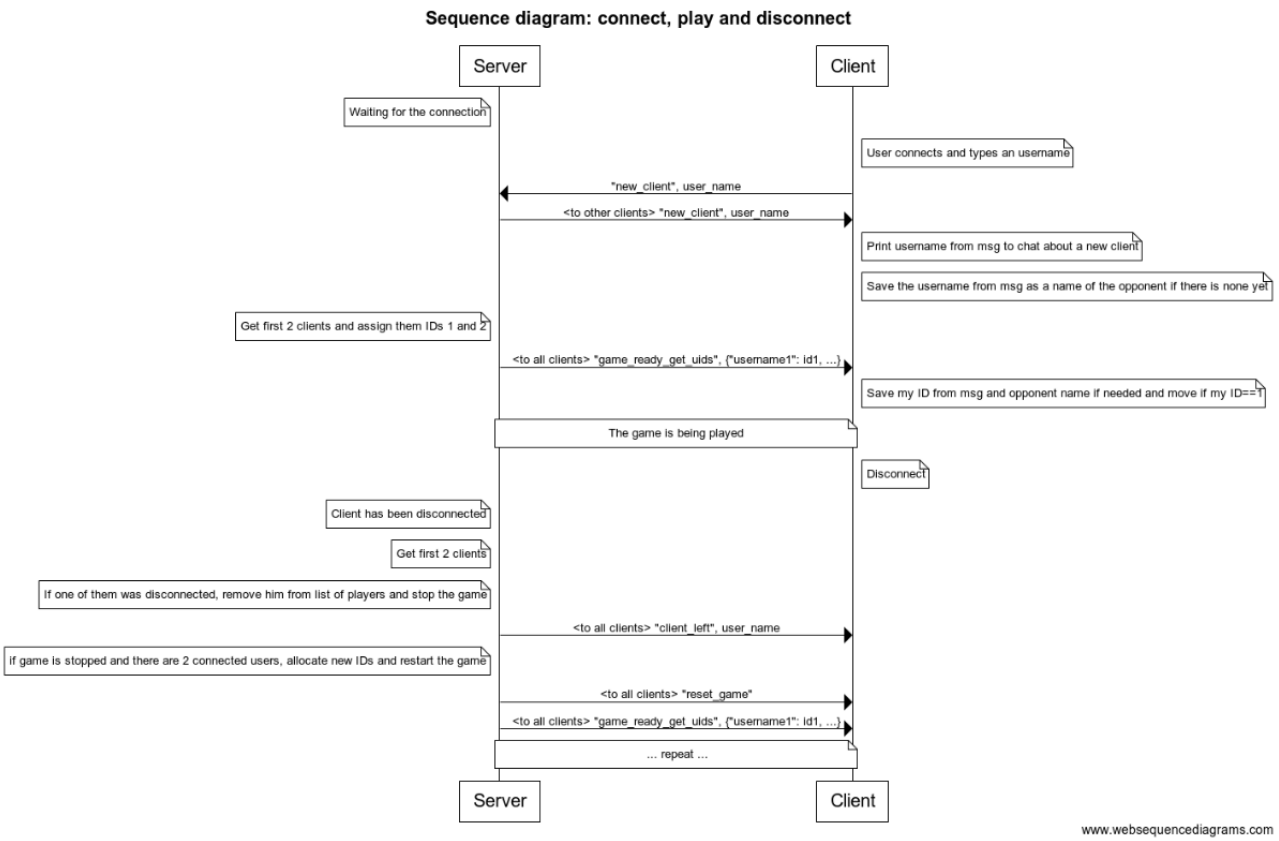

Figure 3: Sequence diagram mainly focused on connecting a new player and later disconnecting.

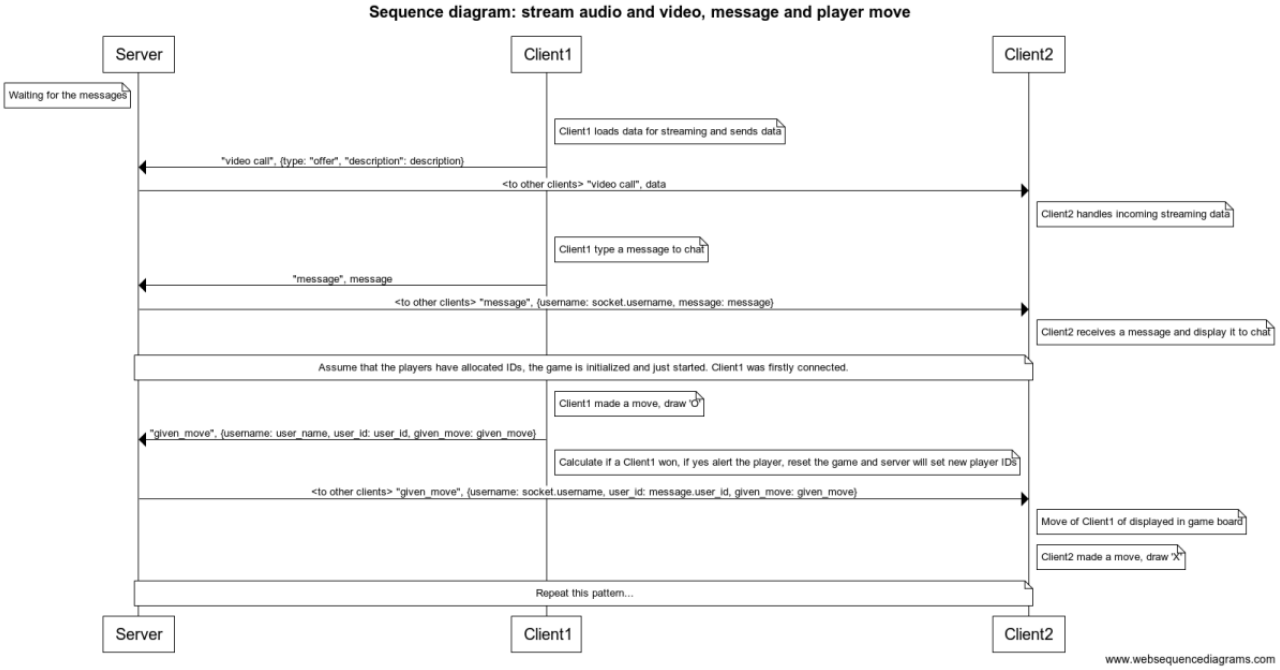

Figure 4: Sequence diagram mainly focused on the game and also chat and audio-video communication.