Movie Synchronization System Using Web Socket Based Protocol

Amar Shukla[1(✉)], Thipendra Pal Singh[1(✉)], Vikas Mishra[1], Garima Goyal[1], Ishita Kanwar[1], Gauraang Sharma[1], and Tanupriya Choudhury[1,2,3(✉)]

[1] School of Computer Science, University of Petroleum and Energy Studies (UPES), Dehradun 248007, Uttarakhand, India
{ashukla,tpsingh}@ddn.upes.ac.in
[2] Graphic Era Hill University, Dehradun 248002, Uttarakhand, India
[3] Graphic Era Deemed to be University, Dehradun 248002, Uttarakhand, India
tanupriya@ddn.upes.ac.in, tanupriyachoudhury@gehu.ac.in, tanupriya1986@gmail.com

Abstract. On a global scale, the Web is a platform that allows universal reach via a variety of communication channels. Whenever there is a newly developed multimedia system or a newly designed web site available, it prevents people from losing track of information that has been shared. Moreover, online streaming, which is a billion-dollar industry, offers a unique opportunity to recast viewing of video from a passive activity into a socially engaged activity by transforming it from a passive event into an online network. Our motivation for developing a multimedia system is to create a movie sync system that allows the user to utilize their own experiences. For the development of this synchronization system, we used the web socket protocol, which allows us to create a collaborative control system over the screen. Due to this, a user can use the prebuffer approach for any multimedia content on the web and can use less bandwidth.

Keywords: Movie Synchronization · Web Socket Protocol · Web Application · Client Server Model

1 Introduction

With fast growing technology, our requirements for the internet have changed from just accessing information to recreating a shared experience [1]. Collaborative video viewing is one such domain that uses capabilities of the internet, but suffers at the hands of bandwidth limitations. Now-a-days, a mass of movies and shows available on the internet, content overload is a serious problem that users encounter while figuring out their watch-list [2].

Moreover, passively streaming content from online platforms can kill time, not boredom. Furthermore, low network bandwidth can append to these problems by causing slow buffering and jerky video playbacks [3]. To bring about a change, a system is required that can allow a group of people to watch videos by overcoming the above mentioned limitations and watch videos together in synchronization [4].

A. Shukla, T.P. Singh, T. Choudhury—All authors are contributed equally.

S. Nandan Mohanty et al. (Eds.): ICISML 2022, LNICST 470, pp. 222–231, 2023.
https://doi.org/10.1007/978-3-031-35078-8_19

This paper focuses on a movie synchronization system that can overcome the stated limitation, allowing users to play pre-downloaded videos in sync at their respective screens [5]. Any action performed by one user, like play, pause, seek or change in audio controls, is reflected at other users' end as well [6].

It is based upon a client server model. Traditionally, this model uses a request response cycle to serve the content to the client browser from the server using Hypertext Transfer Protocol [7]. But it is not fit for processes like synchronization as the real time stream cannot be shared effectively. This project makes use of the web socket protocol as it is very effective for real time communications [8]. Each client establishes a socket connection with the central server endpoint. Any action performed at the clients end is shared to the server which in turn advertises it to all the other clients to establish synchronization [9].

It is based on a client-server model. Traditionally, this model uses a request-response cycle to serve the content to the client browser from the server using Hypertext Transfer Protocol. But it is not fit for processes like synchronization as the real-time stream cannot be shared effectively. This project makes use of the web socket protocol as it is very effective for real-time communications. Each client establishes a socket connection with the central server endpoint. Any action performed at the client's end is shared with the server which in turn advertises it to all the other clients to establish synchronization.

2 Motivation

In an era of continuous change, people are bound to feel stressed and exhausted. To embrace this change, people need company. Building a video synchronization system is an effort to bring people together with technology and that too, using minimal resources. It aims to create a recreational environment virtually, where a group of people can choose and watch movies in synchronization. Moreover, this system can target any locale, the establishes a need for collaborative streaming, be it from educational perspective or recreational domain.

3 Related Work

In order to facilitate remote sharing and synchronization between sites Does not exist "Synchronization requirements for online video viewing in collaboration" discusses the synchronization requirements for watching online videos together [9]. There is an annoyance associated with the synchronization differences between online and offline videos [10]. There was a time lag of 15–30 s between the master and slave computers. The master would send all updates, and the slave would receive all updates from the master [11].

The use of video conferencing in distance learning is described as the next advancement in electronic communication [12]. Video and data signals are transmitted electronically to enable simultaneous interactive communication [13]. Integrated digital systems networks and internet protocol networks are used to process video conferencing data; however, video conferencing data is competing with other computing data with low bandwidth [14].

Researchers in Research on Audio-Video Synchronization Coding based on mode selection in H.26 proposed a way to take advantage of the different modes used by the H.264 encoder during the inter prediction stage to encode audio and embed the audio into the video, and then synchronization coding is applied to combine audio and video. However, the method affected audio and video quality [15].

In the discussion, a virtual theater was mentioned as a synchronization platform. It allows its clients to watch video and chat simultaneously [17]. Guests just have to join the room and enjoy the video, regardless of whether it is from an OTT channel or online. The combination of chatting and watching, however, causes voice lags [16].

While real-time data exchange is frequently required, the Internet's hyperlinked, stateless information flow can be problematic. WebSocket is a protocol for transmitting data between web servers and clients over the internet in an efficient and dependable manner. The authors created a Web application to compare the one-way transmission latency of sending 4 Hz of real-time wind sensor data over WebSocket vs HTTP polling. They created a Jetty servlet to convert an existing HTTP connection to a WebSocket connection. The latency of the WebSocket protocol is compared here to that of HTTP polling and long polling [15].

A more adaptive communication architecture that permits asynchronous messaging between clients and servers and server-initiated data delivery is required for modern Web services. The Websocket protocol and API are essential for developing full-duplex asynchronous Web streams and other Web-based asynchronous communication paradigms. This investigation compares Websocket vs TCP speeds (TCP). In this section, we compare latency and traffic. Except for initial handshaking, websocket-based communication requires no more network capacity than TCP-based communication. Latency is an issue [18].

IoT links consumer electronics. New applications are emerging to make daily life more practical and connected by sharing, collecting, and analyzing personal data. The same technology has the potential to transform how information is handled in industry. The increasing significance of processed data points in Industry 4.0. This essay evaluates two consumer ICT protocols that have significant industrial potential. Experiments were carried out to determine the round trip time between an Italian and a Brazilian server utilizing the MQTT (Message Queuing Telemetry Transport) and WebSocket protocols. Each application and network route may be assessed separately. The request payload size was determined by the average delay samples for each protocol, which varied [19].

IoT applications are growing in popularity. Real-time applications require a low-latency protocol. Using the ESP8266 (SoC with IEEE 802.11) and Node.js servers for data exchange, this work compares the application layer network protocols Message Queue Telemetry Transport and WebSocket, considering documentation, round trip times for packages over a local network, and device memory for the most popular protocol implementation found on Github. Both server and ESP8266 packets were compared to assess latency. WebSocket is superior to MQTT in 1 ms RTT applications, according to the paper [20].

HTML5 WebSocket increases real-time browser communication. Daily online updates occur. WebSockets help. Client-server has been replaced by WebSockets, a new technology supported by current browsers. WebSockets' speed comes from being

duplex and not using HTTP. This article discusses WebSocket server implementations. This section discusses heterogeneous implementations of OpenCL. Real-time Socket Cluster is an all-core WebSocket server [21].

WebSocket is a protocol that allows two-way, real-time communication between a web server and its clients, which are often browser apps. Even though this upcoming standard is supposed to make server and client performance better than traditional XHR-based communication, not much research has been done on the subject thus far. Here, we showcase a WebSocket benchmark framework developed for gauging server performance. Black-box server-side measurements are made possible by this technology. Here, three WebSocket measurement scenarios are used to verify the given architecture [22].

Electricity use worries families today. No way to monitor energy use. Energy use is uncontrollable. The research aims to develop a prototype system to solve these problems. The prototype system lets users see how much power they've used and control appliances online. WebSocket enables real-time Raspberry Pi communication. On-screen admin, app control, and monitoring. A user-friendly system has satisfied customers. Wi-Fi and current sensors may be used in future power monitoring [23].

4 Methodology

Movie synchronization system generates a viable solution for synchronizing movies or other pre-downloaded video content and controls among a group of users connected over the web, overcoming the issue of low bandwidth streaming. This paper focuses on the use of web socket protocol as a means of communication because it is very effective for establishing the real time communications. Each client establishes a socket connection with the central server endpoint. Any action performed at the client's end is shared to the server which in turn floods it to all the other clients to establish synchronization. This functionality is explained below in detail. The architecture of the full process is showing in Fig. 1.

5 Result and Discussion

It is the central server that serves the purpose of synchronization among the clients.

The server is hosted on the Heroku cloud platform and it establishes a socket connection with the client, using the port that is exposed while hosting the application on cloud. The synchronization functionality depends on web sockets. Initially, a room is created for a group of users to interact. For the room creation a random id is generated by the server that is used as the room code.

The server sets this incoming request from the client server through the web socket. It opens a port to listen to the client server and responds to the client with the same. After the connection with the client server is established, it listens to the port for any change in the state of the video being played. On receiving a state change from pause to play or vice-versa, the central server emits the state to all the clients using io.emit(), for reflecting state change at their respective ends. Similarly, if a seek operation is performed at the clients end, the server logs the seek time of the video and advertises it to all the clients to maintain synchronization.

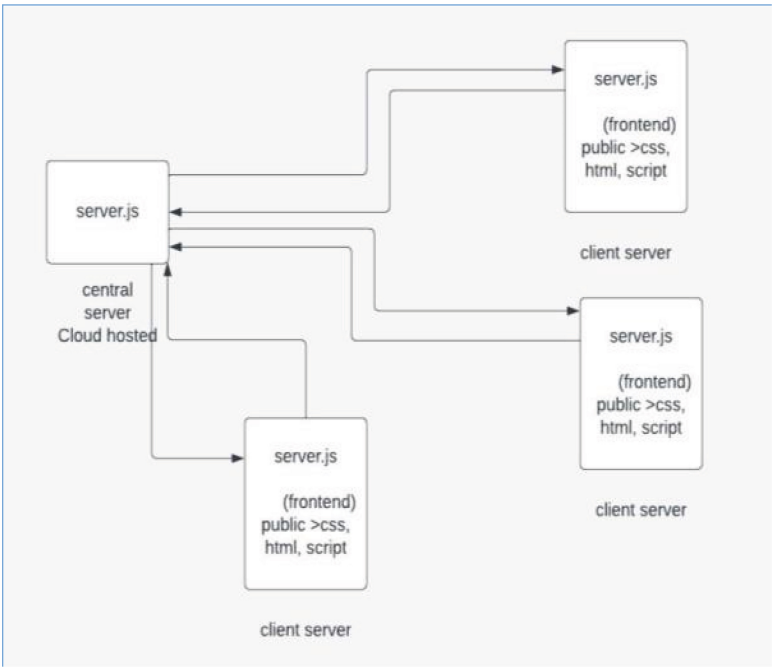

Fig. 1. Architecture of Entire Process

5.1 Movie Client

The client server runs locally on each client's end. It communicates with the central server using socket.io. Client Front end, showing in Fig. 2.

Initially, the user interacts with the client's frontend and can choose to either create a room or join a room. On creating a room, the client receives an id from the central server that acts as a room code and needs to be shared with the other users. If a user wants to join a room, the central server expects the room code to be entered from the client's end and it permits or denies the entry based upon the authenticity of the code.

Once connected to the room, each client establishes its individual socket connection with the central server. This socket connection is used for one to one communication between the client and the server. The client will emit any state change to the central server using socket.emit() which in turn emits it to all the clients in the room.

5.2 Client Server

In order to play the video in synchronization at each user's end, it must be downloaded at their local system. The path of the video is specified in the client server. The server gets the request from the client's frontend for the video stream. To create a stream, the client server divides the video into small chunks of certain size, which are served upon each request from the client's frontend.

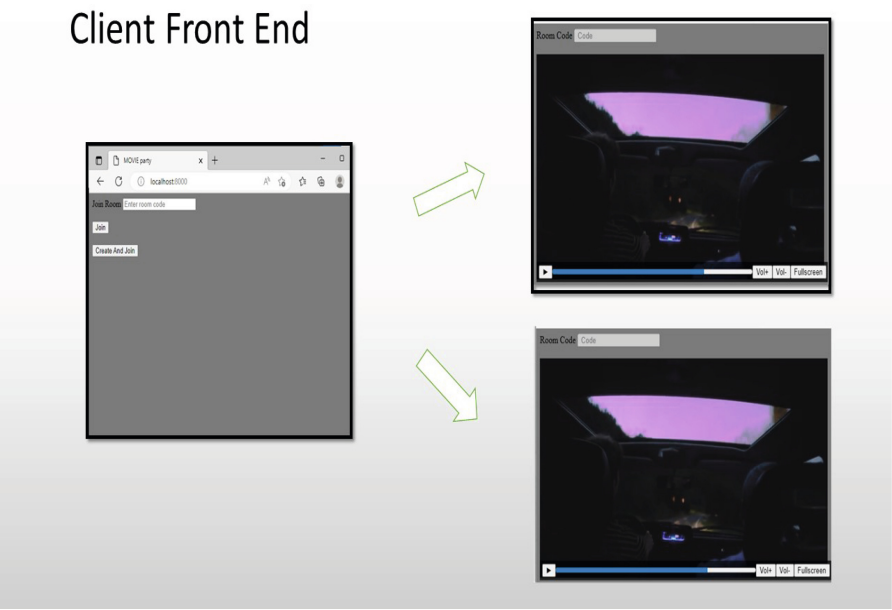

Fig. 2. Client Front End

5.3 Create or Join Room

A message "create-room" is sent through socket communication by the client to the synchronization server. The server creates a 10 character room id and stores it as room Object. The server emits "room-created" message and the room code to the client. The client on receiving creation message further sends the message to join-room along with room Code. The server checks if a room with the provided room Code exists and allows or denies connection based on authenticity. The screenshot of pseudo code of joint room is depicted in Fig. 3.

5.4 Stream Video Locally

On joining the room, the local client frontend sends a request to local server over "/video" to fetch video. The server initializes maximum chunk size of video to be send to 1 MB, calculates the size of video, start time and end time along with the content length. The local server then streams the chunk of the video to the client frontend. And pseudo code for Streaming Quality and Synchronization is depicted in Fig. 4 and Fig. 5 respectively.

5.5 Synchronization

The client's frontend communicates bidirectional with the synchronization server via sockets. If a user clicks play/pause button on the frontend, a message "vid-state" along with room Code as payload is sent to the synchronization server over sockets. The server on receiving "vid-state" message emits the "vid-state" message to all the client devices within the room as that of the user who initiated the event.

```
function handleJoinRoom(data){
            const roomCode = data.roomCode;

            // Check whether the room exists
            if(rooms[roomCode]){
                socket.join(roomCode);
                let roomObject = rooms[roomCode];
                socket.emit("joined", roomObject);
            }
            else{
                socket.emit("message",`The room
${roomCode} doesn't exist`);
            }}
```

Fig. 3. Screen shot of Pseudo code for Joint Room

```
// Setting Headers
          const headers = {
              "Content-Range": `bytes
${start}-${end}/${videoSize}`,
              "Accept-Ranges": "bytes",
              "Content-Length": contentLength,
              "Content-Type": "video/mp4",
          };
```

Fig. 4. Screen shot of Pseudo code for Streaming Quality

```
socket.on('vid-state', data=>{
        const vidState = data.vidState;
        const roomCode = data.roomCode;

io.to(rooms[roomCode].code).emit('vid-state',vidState);
    })
```

Fig. 5. Screen shot of Pseudo code for Synchronization

6 Conclusion

The popularity of this application stems from the fact that most people stream videos and movies on online platforms. It is possible to host groupware for movie synchronization to allow people throughout the world to see the same content at the same time at different locations. In order to achieve synchronization, everyone at any given time would be watching the same part of the video. The web socket protocol is at the core of the synchronization module because it allows clients to communicate event changes in real time across all devices. The pre downloaded videos on each client ensure that the entire functionality can be accessed even when bandwidth is low. The usefulness of this application stems from the fact that people often stream videos and movies on online platforms. Hosting groupware for movie synchronization can allow a group of people to watch the same content at the same time but from different locations. Synchronization would ensure that everybody is watching the same part of the video at any instance of time. The web socket protocol serves as the backbone of the synchronization module as it enables the client to communicate event changes in real-time across all the devices. Pre-downloaded videos on each client system ensure that the entire functionality is supported even at a low bandwidth.

References

1. Ijaz, U., Quacchio, E., Alfonso, D.: A web-based framework for compressed 3D objects: downloading and rendering. In: 2012 10th International Conference on Frontiers of Information Technology, pp. 129–133. IEEE (2012)

2. Popescu, V.G., Burdea, G.C., Bouzit, M., Hentz, V.R.: A virtual-reality-based telerehabilitation system with force feedback. IEEE Trans. Inf Technol. Biomed. **4**(1), 45–51 (2000)
3. Salehi, M., Alipour, M.: E-banking in emerging economy: empirical evidence of Iran. Int. J. Econ. Financ. **2**(1), 201–209 (2010)
4. Redfern, S., Galway, N.: Collaborative virtual environments to support communication and community in internet-based distance education. J. Inf. Technol. Educ. Res. **1**(1), 201–211 (2002)
5. Gül, S., et al.: Cloud rendering-based volumetric video streaming system for mixed reality services. In: Proceedings of the 11th ACM Multimedia Systems Conference, pp. 357–360 (2020)
6. Zhu, G., Zhang, F., Zhu, W., Zheng, Y.: HTML5 based media player for real-time video surveillance. In: 2012 5th International Congress on Image and Signal Processing, pp. 245–248. IEEE (2012)
7. Gül, S., et al.: Interactive volumetric video from the cloud (2020)
8. Quax, P., et al.: Remote rendering solutions using web technologies. Multimedia Tools Appl. **75**(8), 4383–4410 (2015). https://doi.org/10.1007/s11042-015-2481-0
9. Prins, M., Niamut, O., van Brandenburg, R., Macq, J.F., Rondao Alface, P., Verzijp, N.: A hybrid architecture for delivery of panoramic video. In: Proceedings of the 11th European Conference on Interactive TV and Video, pp. 99–106 (2013)
10. Iacono, L.L., Guillén, S.S.: Efficient and adaptive web-native live video streaming. Int. J. Adv. Internet Technol. **7**(3 & 4), 2014 (2014)
11. Salminen, S.: Improving the performance of a couchDB powered websocket API in low-quality internet conditions (2019)
12. Varisetty, T., Dietrich, D.: Client-side bandwidth estimation technique for adaptive streaming of a browser based free-viewpoint application. In: Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges, pp. 39–44 (2019)
13. Liu, Q., Sun, X.: Research of web real-time communication based on web socket (2012)
14. Lombardi, A.: WebSocket: Lightweight Client-Server Communications. O'Reilly Media, Inc. (2015)
15. Pimentel, V., Nickerson, B.G.: Communicating and displaying real-time data with websocket. IEEE Internet Comput. **16**(4), 45–53 (2012)
16. Skvorc, D., Horvat, M., Srbljic, S.: Performance evaluation of websocket protocol for implementation of full-duplex web streams. In: 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1003–1008. IEEE (2014)
17. Tiwari, S.: An ensemble deep neural network model for onion-routed traffic detection to boost cloud security. Int. J. Grid High Perform. Comput. (IJGHPC) **13**(1), 1–17 (2021)
18. Skvorc, D., Horvat, M., Srbljic, S.: Performance evaluation of websocket protocol for implementation of full-duplex web streams. In: 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1003–1008 (2014). https://doi.org/10.1109/MIPRO.2014.6859715
19. Silva, D.R.C., Oliveira, G.M.B., Silva, I., Ferrari, P., Sisinni, E.: Latency evaluation for MQTT and websocket protocols: an industry 4.0 perspective. In: 2018 IEEE Symposium on Computers and Communications (ISCC), pp. 01233–01238 (2018). https://doi.org/10.1109/ISCC.2018.8538692
20. Oliveira, G.M.B., et al.: Comparison between MQTT and websocket protocols for IoT applications using ESP8266. In: 2018 Workshop on Metrology for Industry 4.0 and IoT, pp. 236–241 (2018). https://doi.org/10.1109/METROI4.2018.8428348
21. Muller, G.L.: HTML5 WebSocket protocol and its application to distributed computing. arXiv preprint arXiv:1409.3367 (2014)

22. Imre, G., Mezei, G., Sarosi, R.: Introduction to a websocket benchmarking infrastructure. In: 2016 Zooming Innovation in Consumer Electronics International Conference (ZINC), 84–87 (2016).https://doi.org/10.1109/ZINC.2016.7513661
23. Miu, A., Ferreira, F., Yoshida, N., Zhou, F.: Generating interactive websocket applications in typescript. arXiv preprint arXiv:2004.01321 (2020)