iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** XII **Month of publication:** December 2023

DOI: https://doi.org/10.22214/ijraset.2023.57717

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue XII Dec 2023- Available at www.ijraset.com

WebRTC: Revolutionizing Communication in the Digital Era

Saifullah Rahman[1], Shivam Kumar[2], Sanaul Mustafa[3], Mansi Gupta[4], Prachi goel[5], Apurva jain[6]
CSE Department, ADGIPS

Abstract: ***Web browsers are equipped with Real-Time Communications (RTC) capabilities thanks to WebRTC, a peer-to-peer technology. The majority of contemporary browsers incorporate this technology with ease, allowing them to speak with one another directly instead of only depending on web servers. Once connectivity is established, browsers may share files or send messages using peer-to-peer networks, which is the most effective way to exchange media streams, including audio and video from microphones and cameras. By utilizing this fundamental peer-to-peer technology, web apps are able to provide a variety of learning experiences. With WebRTC handling traffic and peer-to-peer media stream relaying, learners may find specialists and make direct contacts. The system also supports file sharing, screen sharing, and messaging, which enhance learners' ability to actively engage with experts to get a better understanding of certain ideas. Web browsers are so common on PCs and mobile devices that they have contributed to the broad acceptance of the web application paradigm.***
Keywords: WebRTC; peer-to-peer; education; online learning; communication; media.

I. INTRODUCTION

Online learning is a hugely revolutionary breakthrough in the field of modern education. A new age has been brought about by the broad adoption of network education, which has been fueled by developments in computer technology and the internet. Activities related to teaching and learning are made simpler by the internet's integration with education. When educators provide instruction over the internet and pupils study online, data is exchanged, knowledge is gained, and offline activities eventually become an adjunct to online pursuits. These days, students are not limited to traditional classroom settings; instead, they may access content at their own pace by using online learning resources. A well-liked and useful technology that provides a web-based learning environment is virtual learning. Currently, instructors and students have access to proprietary technology for audio and video network education. But these systems frequently need the installation of standalone apps or extra plug-ins, like the widely used Adobe Flash plug-in and the well-known VoIP program Skype. Regrettably, customers may find these solutions to be cumbersome due to lengthy setup and installation processes; some even charge a registration fee.

WebRTC has changed the game by enhancing the interactivity and immersion of virtual learning. The features of WebRTC facilitate the establishment of a virtual learning environment by offering a smooth and effective platform for media exchange and communication within the online education domain.

Learning systems may be utilized by institutions and organizations to optimize WebRTC's capabilities, providing a plethora of sophisticated features and improving user experiences. It is therefore a priceless resource for anybody looking for an immersive learning environment.

"Web Real-Time Communications," or WebRTC for short, is a powerful and state-of-the-art technology that supports virtual learning environments. WebRTC's foundational technology has been standardized for the transmission of audio and video files.

With its foundation in HTML5, the WebRTC-based peer-to-peer learning system has all the necessary tools to enable real-time audio and video communication via a web browser without requiring extra plugins or third-party software such as Java or Adobe Flash. Without the need to install plugins or configure software, this system effortlessly connects a number of services, such as audio/video conferencing, presence, and instant messaging. These educational services' accessibility and usability play a major role in the general public's decision to accept them. Online learning systems based on WebRTC have a wide range of applications and substantial development potential.

A. Overview

WebRTC enables peer-to-peer connections between web browsers to be established with ease. In contrast to conventional approaches, which involve integrating many frameworks and libraries to handle problems like packet loss, connection dropouts, and NAT traversal, WebRTC integrates these features directly into the browser.

©IJRASET: All Rights are Reserved

Notably, it is open-source and does away with the requirement for plugins or other third-party software, which simplifies the development process. WebRTC is useful for many things, but its main benefit is peer-to-peer multimedia (audio and video) communications in real time. A mutual consent to start a conversation, location awareness, getting beyond firewalls and security measures, and real-time multimedia content transfer are all necessary for users to communicate via web browsers. The microphone and video camera on desktop or mobile devices may be accessed by browser-based apps thanks to WebRTC. When an application asks to access one of these devices, users are usually told, and WebRTC creates separate streams of transmittable audio and video data after granting access. After then, this data is sent via bidirectional network data channels, enabling peer-to-peer information sharing.

B. Objective

The main objective of the peer-to-peer learning system is to help students learn by interacting with professionals; WebRTC technology is used to make this process easier for students to complete. WebRTC will greatly ease the load on end users in 2022 by doing away with the requirement to install extra plugins like Adobe Flash and outside applications.

WebRTC is a peer-to-peer technology that eliminates the need to manage a central server in favor of decentralized peer connections. Because data flow is immediately shared between users without the need for middle servers, the decentralized structure of this system guarantees user privacy. In addition, the system seeks to promote dynamic exchanges via file sharing, screen sharing, and messaging, improving efficient communication between users.

II. RELATED WORKS

The focus of the standardization efforts proposed by IETF and W3C, as described in [1], is on web-based real-time multimedia applications. An architecture that includes a full suite of protocols has been developed by the RTCWEB working group to enable dependable real-time multimedia communication across web browsers. The presentation covers the ongoing work on the WebRTC 1.0 standard and introduces the principles of ORTC (Object Real-Time Communication) that will be incorporated into future standardization.

The WebRTC system's general architecture is described in depth in [2], with a focus on the peer-to-peer exchange of data between two web browsers. The media transport and safe encryption are covered by the protocols listed. WebRTC data channels are explained in detail, and transport layer problems like firewall traversal and NAT are covered.

The effects of WebRTC as a corporate application are examined in [3]. Peer-to-peer data transfer, firewall traversal, access control, and other corporate use challenges are covered. Along with highlighting the changes brought about by WebRTC within businesses and in external engagements, integration and interoperability challenges with the current communication infrastructure are also addressed.

[4] discusses the packet loss that occurs in the network along the video transmission channel. A hybrid NACK/FEC technique is suggested as an adaptive system to balance end-to-end latency, seamless rendering, and video quality. It is proposed to use an offline simulation tool to analyze system behavior under different network circumstances.

A list of possible WebRTC security flaws may be found in [5]. Due of its open nature, WebRTC is vulnerable to network-based assaults that might have serious security repercussions. While HTTPS may be sufficient for security in standard applications, more intricate connections call for a more reliable solution. In order to ensure confidentiality and data integrity, the suggested security architecture uses HTTPS and TLS (Transport Layer Security) to create a secure session with a server that has valid credentials.

III. EXISTING SYSTEM

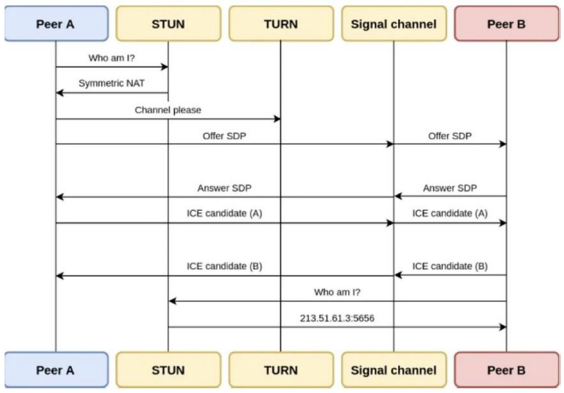
The current system operates inside the boundaries of institutions with proprietary features and depends on document-based or web-page-based user interfaces. For participant communication and coordination, it uses a centralized server, which adds significant maintenance and handling overheads that drive up expenses. Furthermore, the pay-per-use nature of this system means that it is unavailable to the general public, which raises user costs even more. The complex development process necessitates the installation of proprietary plugins and third-party software, such as Adobe Flash, which raises use expenses. These systems take a while to set up, some need registration fees, and more client-side storage is required. Users have a steep learning curve, and dynamic participant interactions are not supported by the systems. The existing systems have several shortcomings, including the inability to enable one-on-one conversations with specialists, the requirement for separate plugin and third-party software installs, and their restriction to institutional usage. Their reliance on centralized servers compromises data security by introducing communication overhead and raising privacy issues because all relayed traffic goes through a server.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue XII Dec 2023- Available at www.ijraset.com

IV. PROPOSED SYSTEM

By utilizing the underlying WebRTC technology, the system seeks to expedite the process of sharing information by promoting efficient communication between professionals and students. The WebRTC's no-plugin method greatly reduces setup time by not requiring users to use third-party applications like Skype or install plugins like Adobe Flash. The web application is retrieved from the Heroku Cloud server using HTTP requests, and users may access it by using a URL in their web browsers. Every user, whether a student or an expert, registers in and submits information, such as the expert's field of expertise. Students verify their information and ask for professional assistance after logging in. The system connects when an expert becomes available, enabling the call to continue and multimedia material to start flowing. Experts indicate when they are available for a connection by entering their verified information and area of expertise. The expert functions as the other peer in the network and the system creates connectivity if a student is waiting on the other end. Users on the network may exchange knowledge in an easy and effective manner thanks to this method.



Multimedia data is sent between the two peers while call setup takes place. Participants can engage dynamically with the provision of additional features including file transfers, screen sharing, and messaging. The user only has to submit the website URL to establish communication with the other peer thanks to the system's user-friendly user interface, which makes it simple to grasp. The suggested approach illustrates WebRTC's learning power by enabling the use of voice, video, and message to highlight certain issues you are worried about or to make sure the person on the other end understands exactly what you are saying.

A. Advantages

1) Direct real-time communication with specialists is made possible by the system, which was not possible with the previous one. Students are able to learn and go on appropriately.
2) Enables dynamic interactions: In addition to supporting sophisticated functions like screen and file sharing for more active involvement, the system allows users to engage dynamically using media data like voice and video.
3) Easily accessible for public use: Because the system is housed on a public server, end users may easily access and utilize it.
4) Decentralized approach: It is not economical to maintain the current method, which entails keeping a separate server to handle media traffic. As the connectivity is created directly between the peers, the suggested solution does away with the requirement for a server.
5) Plugin-free mechanism: Users may have a hassle-free experience as the system operates without the need for extra plugins or third-party applications. The total cost is substantially lower because no plugins are needed.

B. Methodology

Private and public IP addresses are the two main categories of IP addresses. Every device in a network is given a private IP address by the router; these addresses are not accessible via the Internet. A public IP address, on the other hand, is allocated to a device upon connection to the Internet and is directed towards the Internet. A device called Network Address Translation (NAT) makes it easier for devices connected to a home network to uniquely identify themselves by translating IP addresses from the public to the private domain.

Session Traversal Utilities for NAT (STUN) is used to facilitate peer-to-peer communication, which necessitates an understanding of each other's IP addresses.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The public IP address is returned by the STUN server in response to a query from the web client. Both clients go through this process to establish connection behind the NAT. The web clients then submit an HTTP request to the Web Application Server (such as Heroku Cloud) in order to access the application that has been stored there.

Session Description Protocol (SDP) is used for communication and session negotiation after peers are connected to the same "channel." The initiating peer waits for a connected receiver to "answer" after sending a "offer" over SDP. Each peer creates local data streams and data channels after getting the response, at which point multimedia data transfer starts. In the application scenario, the student inputs their login information and bides their time for the expert to establish a connection. The specialist logs on with their credentials and area of expertise, starting a peer-to-peer video conference.

V. MODULE DESCRIPTION

A. *Student Details*

By visiting a webpage and providing the required information, the student starts the session. As one peer in the network, the student seeks to communicate with the expert after receiving validation.

B. *Expert Details*

Once the expert's information is verified and his area of expertise entered, he indicates that he is available and the connection may continue. In the network, the expert serves as the other peer.

C. *Call Establishment*

After the learner and the expert input their information on either end, the call setup procedure starts, allowing both parties to communicate via video feeds in real time. The learner and expert create instantaneous video communication. The internet connection speed affects the video quality; faster connections yield better quality. The TURN (Traversal Using Relays around NAT) server is used to redirect video traffic between peers in the event that the public STUN server fails. It is crucial to remember that TURN server maintenance can be quite expensive.

VI. WORKING

The WebRTC Peer to Peer Learning system is made up of several crucial parts, such as:

A. *Stun And Turn*

The goal of the STUN server is to help clients discover their public IP address. This is especially helpful for clients who are behind a network appliance (NAT) and find it difficult to find their public IP address. Using both TCP and UDP connections, STUN functions as a client-server protocol. In response to client STUN requests, a STUN server handles problems arising from non-standard NAT behaviors. The simple role of STUN servers, which are usually located on the public internet, is to verify the IP: port address of incoming requests and reply to the asking clients with the address that was found. In WebRTC calls, publicly accessible STUN servers—such as those provided by Google—are frequently used.. By enabling peers to find each other and establish contact through the signaling method, these servers are essential in enabling IP communication and creating a direct link. Additionally, TLS (Transport Layer Security) protocol encryption is supported by STUN, guaranteeing message integrity and authentication.

When peer-to-peer communication becomes problematic, the TURN server becomes active. Because it can traverse symmetric NATs, TURN has an advantage over STUN. When the STUN server malfunctions, the TURN server steps in as a backup, especially in situations where the STUN servers are unreliable in managing high media traffic volumes or exhibit low dependability. Nevertheless, the TURN server has a major cost, requires a lot of maintenance, and uses a lot of bandwidth, particularly when delivering HD video streams.

B. *Signalling*

Through the use of the Session Description Protocol (SDP), signaling facilitates peer-to-peer exchange of session control messages. It also allows network parameters, such as media capabilities and ICE candidates, to be sent. A method of message exchange between clients is necessary for this signaling process. During the exchange of sessions, one client makes an offer, the other answers, and the connection advances if either client accepts. It's crucial to understand that the signaling mechanism must be built separately from WebRTC APIs.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Socket.IO is used for signaling since it provides a reliable solution. A JavaScript library called Socket.IO offers a dependable way to connect peers in real time. In addition to functioning as the signaling medium, Socket.IO also acts as a fallback method in the event that problems arise with regular WebRTC peer-to-peer communication. It is not necessary to set up a separate signaling exchange or deploy and scale additional servers in order to integrate Socket.IO; this may be done with a straightforward middleware.It only takes a few lines of code to set up the server and client. Server:
var io = require ('socket.io') (server);
Client: var io = require ('socket.io-client');

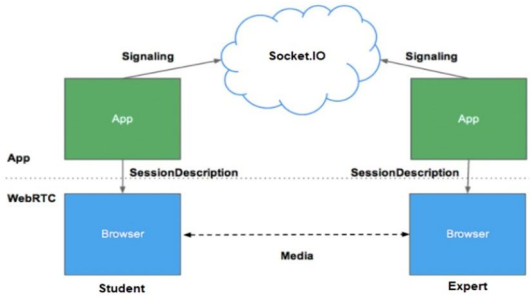


Fig. 3. Signaling

C. Web RTC APIs

1) *Media Stream API:* This API is used to record media streams from nearby cameras and microphones. The getUserMedia() function is the main way to retrieve these local streams using the MediaStream API. Multiple MediaStreamTrack objects make up each Media Stream object. These streams—which include audio and video—come from input devices that are connected. The Media Stream Track object can also include other channels, such the left and right audio channels, which are specified by the smallest units in the Media Stream API.
2) *RTC Peer Connection API:* This is the main part that allows the browsers to connect to each other peer-to-peer. We include the following in order to construct the objects for the API: var pc = (config) RTCPeerConnection; An array of ice servers is stored in a key that is part of the "config" argument. As inputs, an array of URL objects containing information about STUN and TURN servers is provided.
3) *RTC Data Channel API:* This API offers a channel for binary data transfers that are bidirectional and peer-to-peer. A data channel event is sent to the peer that joins the channel in order to exchange data, informing it that a data channel has been added to the connection.

VII. IMPLEMENTATION

1) WebRTC implementation requires a number of important stages and considerations. This is a high-level summary of the procedure:
2) Configuring an environment for WebRTC:
3) Verify that WebRTC is supported in your development environment. WebRTC is supported by popular browsers including Chrome, Firefox, and Safari.
4) Integration of JavaScript and HTML:

- Construct HTML webpages to house your WebRTC application.
- To communicate with the WebRTC API, use JavaScript.

5) . getUserMedia API: You may access the user's camera and microphone by using the getUserMedia API. This is essential for streaming video and audio capture.
6) Constructing a MediaStream: • Merge the video and audio streams into an object called a MediaStream. The media material that will be transferred in real time will be represented by this object.

Setup of Peer Connection:

- Create a peer-to-peer network among users. This entails generating a JavaScript RTCPeerConnection object.

7) Signalling: Implement a signalling mechanism for communication between peers. This involves exchanging session control messages, including offers and answers, and sharing information about network configurations.
8) ICE (Interactive Connectivity Establishment):

- Use the Interactive Connectivity Establishment (ICE) framework to enable communication across different network configurations, including NAT traversal. This is crucial for ensuring connectivity between peers.

9) SDP (Session Description Protocol):

- Implement the Session Description Protocol (SDP) to describe media streams and negotiate capabilities between peers during the session initiation process.

10) Handling STUN and TURN Servers:

- Include STUN (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT) server information to facilitate optimal connectivity, especially in cases where direct peer-to-peer communication is challenging.

11) Handling Data Channels (Optional):

- If required, implement data channels for exchanging non-media data between peers.

12) Security Considerations:

- Implement security measures, including encrypting media streams using Secure Real-time Transport Protocol (SRTP) and ensuring secure signalling.

13) Testing and Debugging:

- Thoroughly test your WebRTC application in various network conditions to ensure robust performance. Use debugging tools as needed.

14) Deployment:

- Deploy your WebRTC application on a server accessible to users.

VIII. CONCLUSION AND FUTURE WORK

Our solution is a peer-to-peer learning system that we designed and put into operation using the underlying WebRTC technology. This strategy increases accessibility and reach while encouraging teamwork and communication, improving engagement both within and outside of the classroom. The main goal is to increase learning's affordability and cost-effectiveness, which is made possible by WebRTC's lack of a plugin. Furthermore, WebRTC's peer-to-peer design has the potential to lower infrastructure and network costs.

Because of its simple JavaScript APIs for handling multimedia data streams, WebRTC makes it easier to develop real-time applications. The current system can only allow one-to-one conversations because of increased resource needs and bandwidth use. Future iterations, on the other hand, plan to facilitate one-to-many interactions, which will enable the system to grow and support several users at once.

Furthermore, the system could incorporate a virtual whiteboard feature, empowering participants to sketch diagrams and images in real-time. This functionality contributes to enhancing the learning capabilities of the system, providing an interactive and dynamic learning environment.

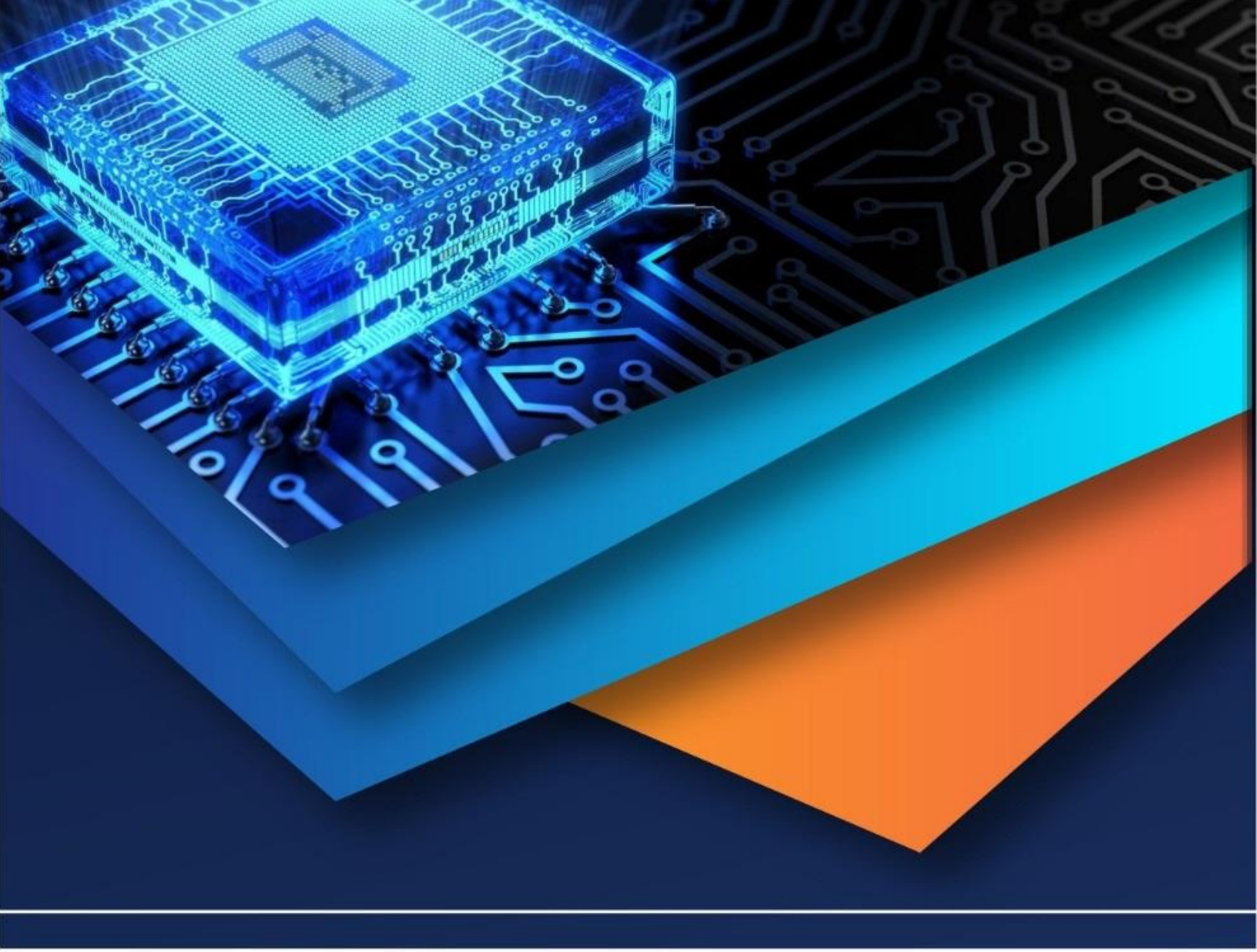
REFERENCES

[1] Felix Weinrank, Martin Becke, et al, “WebRTC Data Channels”, IEEE Communications Standard Magazine, vol. 1, no. 2, July 2017, pp. 28-35, DOI: 10.1109/MCOMSTD.2017.1700007

[2] A. Johnston, J. Yoakum, and K. Singh, “Taking on WebRTC in an Enterprise,” IEEE Communication Magazine, vol. 51, no. 4, April 2013, pp. 48–54, DOI: 10.1109/MCOM.2013.6495760

[3] S. Holmer, M. Shemer, and M. Paniconi, “Handling Packet Loss in WebRTC,” 2013 IEEE Int’l. Conf. Image Processing, Melbourne, VIC, 2013, pp. 60–64, DOI: 10.1109/ ICIP.2013.6738383

[4] R. L. Barnes and M. Thomson, “Browser-to-Browser Security Assurances for WebRTC,” IEEE Internet Computing, vol. 18, no. 6, Dec. 2014, pp. 11–17, DOI: 10.1109/MIC.2014.106



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)