iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** V **Month of publication:** May 2022

DOI: https://doi.org/10.22214/ijraset.2022.43595

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 10 Issue V May 2022- Available at www.ijraset.com

A WebRTC Based Video Conferencing System with Screen Sharing

Abhay Kasetwar[1], Nikita Balani[2], Deepika Damwani[3], Alfiya Pandey[4], Aafreen Sheikh[5], Apeksha Khadse[6]
[1, 2, 3, 4, 5, 6]*Department of Electronics And Telecommunication, Nagpur University/ SB Jain Institute of Technology Management & Research, Nagpur, Maharashtra, India*

Abstract: ***With the modern and rapid development of the internet, people's connections are more important than ever, and they're looking for new ways to communicate with one another in real time. WebRTC is a futuristic technology that enables real-time communication in audio, video, and data transmission through web browsers without the need for a plugin by using JavaScript APIs (Application Programming Interface). In this paper, we present a web peer-to-peer real-time communication system that allows users to communicate across a communication channel with high-speed data transmission using WebRTC technology, HTML5, and a Node.js server address. The outcome demonstrates that the system is stable, fully functioning, and secure.***

Keywords: WebRTC, Vide o Conferencing Application, Peer to Peer.

I. INTRODUCTION

In the current scenario, online learning is without a doubt the most transformative breakthrough. While teachers teach over the internet, students study over the internet, information flows, and knowledge is gained, and eventually, all offline activities will become supplements to online activities. Virtual Learning is currently one of the most well-liked and successful technologies for creating a web-based learning environment. Only proprietary audio and video network education solutions are now available to teachers and students. However, each of these methods requires the installation of an additional plug-in or a completely standalone application, such as Adobe Flash and the most popular VoIP application Skype, both of which are inconvenient. Each of these systems takes a long time to install and configure, and some of them even charge a registration fee. Because of WebRTC, online learning has become even more engaging and dynamic. Virtual learning allows us to spread out our learning, increasing our ability to use approaches that improve the learning process. Institutions or organisations can use the capabilities of such a virtual learning system to guarantee that legacy systems take full use of WebRTC. The most modern and powerful technology supporting virtual learning systems are WebRTC ("Web Real-Time Communications"). WebRTC is a technique for transmitting audio and video material that has been standardised. HTML5 and a set of standards, protocols, and JavaScript APIs are used in the technology. The HTML5 video tag and the get User Media function allow a video stream to travel between two browsers. It allows you to combine diverse services like Instant Messaging, presence, and audio/video conferencing without the need for any plug-ins or setup tools. The development area and possibilities for WebRTC-based online learning systems are vast.

II.TECHNOLOGIES USED

1) *WebRTC:* Web Real Time WebRTC: Web Real Time Communication
2) *SDP:* Session Description Protocol
3) *JS:* JavaScript
4) *HTML:* Hypertext Markup Language
5) *CSS:* Cascading Style Sheet

A. *Web Real Time Communication (WRTC)*

Web Real-time Communication is a framework that allows browsers to communicate with each other in real time. WebRTC allows browsers to natively broadcast audio, video, and arbitrary data to one another without the use of a central server. This eliminates the need for plugins or platform-specific apps when writing and running real-time applications like communication services directly in the browser. A voice engine, video engine, and transport and communication technologies are all included in WebRTC. It's utilised for conferencing, social networking, online medical consultations, live games, and other applications that require Real Time Communication (RTC). It allows developers to create RTC Data Channel, RTC Peer Connection, and Media Stream objects.

1) Background of WebRTC Technology

We can make real-time video conversations to folks on the other side of the country, swap music and data with our friends, even listen to lions screaming in Africa in real time, all thanks to WebRTC technology. These technologies, however, are not freely available. WebRTC has JavaScript APIs and a protocol. regulations, and encryption principles that we can use to create an Real-time communication application. WebRTC has three APIs as well as numerous additional core components such as an encryption system.

STUN/TURN servers, a signalling server, ICE, SDP, NAT, UDP, and a signalling server TCP and UDP are used to establish
TC direct peer-to-peer connections. Next, we'll look at each of these elements individually.

The three JavaScript APIs are (a) the get User Media API, which allows a browser to access audio and video streams; (b) the RTC Peer Connection API, which handles major communication tasks as well as media stream exchange; and (c) the RTC Data Channel API, which allows data to be transferred from one peer to another.

2) getUserMedia() API

If we wish to capture an audio or video stream on a computer before HTML5 is implemented, we must use flash or JavaScript-based plugins. HTML5 now includes this device accessing capabilities, and video and audio are integrated into browsers. The browser has the ability to

specifying video, you can gain access to a user's camera and microphone. on the HTML5 page, as well as audio tags

these tags are specified in the method navigator of the video chat WebRTC application. getUserMedia(), which requires two arguments additional parameters to handle what to do when the video/audio streams are properly collected, and what to do when they are not

if it is unable to capture them When using this application in a browser that specifically requests permission from users,

It is acceptable to utilise a webcam. Encryption is one of the operations that are required for all layers of the protocol. If messages transferred between browsers are stolen or hijacked in the middle of transmission, they are unreadable. In the Security section, we'll go over these encryption features in further detail. Regardless, have a half-decent code in the navigator to specify video/audio streams. This function, at most, getsUserMedia() alone is insufficient to see the user's face. It has to be RTCPeerConnection registered ()

FLOWCHART

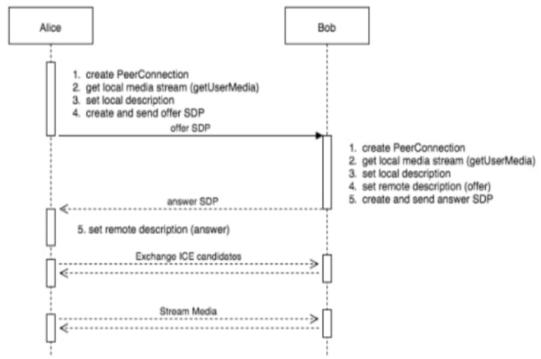


Fig 1 Flowchart of WebRTC

3) RTC Peer Connection API

The RTCPeerConnection object is in charge of the components involved in connecting two browsers to share real-time media. It establishes a connection and collects gICE candidates, which entails obtaining the public IP address and port address of two browsers.

When two browsers want to share data, they need to be able to exchange three different types of information. They must first select whether or not to link. close by going over the session control data They must then figure out how to exchange network data like each peer's IP address and port numbers. Finally, they'll have to know how to deal with media data like the codecs and media formats used by the peers in the connection.

We provide a rough overview of how the RTCPeerConnection API can be utilised in this section. This API actually uses a lot more events, methods, and properties than are listed here. This API is accompanied by a number of underlying protocols, which we shall discuss in greater depth in this section.

The STUN and TURN servers required to locate the ICE candidates are included in the parameter used to generate the RTCPeerConnection object. Google offers free public STUN servers at code.google.com, however there aren't many free TURN servers, and the only ones that are trustworthy are commercial A caller (peer) must first construct a new RTCPeerConnection object called pc. Now pc must construct an ICE candidate by calling onicecandidate(), which returns a list of ICE candidates returned by STUN/TURN servers, each of which contains the current browser's public IP address and port. The signal server transmits these numbers to the target peer. The getUserMedia function then receives a local and remote media stream for the pc object. This For a remote media stream, the onaddstream() method must be used to attach it to RTCPeerConnection.

For a local video/audio stream, use the addStream() method. The caller pc must use the createOffer method to create an offer () method. This package includes codecs, encryption algorithms, and more. as well as any candidates acquired by the ICE agent, All of these are encapsulated in SDP, which is supplied as a parameter to a new RTC Session Description (offer) object. The RTCPeerConnection is then used by PC. setLocalDescription() method for sending data to a target peer via a signalling server

B. Session Description Protocol (SDP)

In the last section, we examined the relevance of a signal server in a WebRTC application and the vital role it plays in data exchange between two peers; however, the signal server cannot work alone. To execute its purpose, it requires the help of numerous other underlying protocols, one of which being SDP. SDP's major function is to share media-related information based information through a network with other peers SDP provides the session's name, purpose, media type, protocols, codec and its settings, timings, and transportation information. When the RTCPeerConnection object begins collecting ICE candidates for establishing a connection with another user, an SDP description is constructed. SDP has been present for a long time (since the late 1990s) for media-based connections, and it has gained experience through a variety of different uses, such as phones, before it was used for the first time. WebRTC will rely significantly on it. The SDP format is text-based. With a line breaker at the end, it consists of a set of key-value pairs. This is an example: "key>=value>n." This key uses a single character to represent the value type, with a machine-readable configuration value is the value. It use mnemonic names like the ones listed below.

III. LITERATURE SURVEY

There have been many researches that explored the architectural choices for internet broadcast. In [2], the authors reviewed the architectural choices for supporting Internet broadcast.

In the Internet environment, the major issue for broadcast is determining the layer of implementation. There are two conflicting considerations 1) a functionality should be pushed to higher layers if possible unless 2) implementation at a lower layer can achieve significant performance benefits that outweighs the additional complexity. In [3], the authors argued

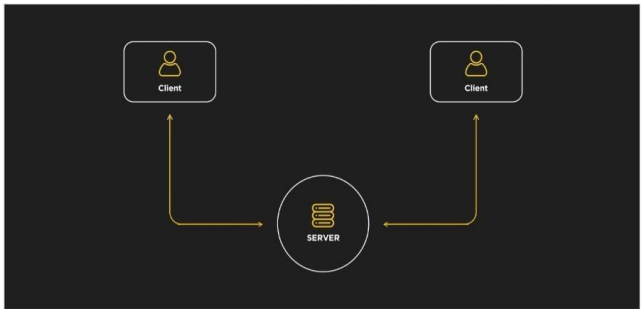
that this second consideration should prevail and it has since been widely accepted, leading to the IP multicast model of today. Unfortunately, despite the tremendous effort in the past today’s IP multicast deployment remains limited. Three complex reasons are making this approach less attractive. First, IP multicast requires router support which introduces high complexity and serious scaling constraints. Secondly, IP multicast attempts to conform to the traditional separation of routing and transport that has worked well in the unicast

context. However, providing higher level features is proved to be more difficult than in the unicast case. Finally, IP multicast calls for changes at the infrastructure level, and this slows down the pace of deployment.

In the new millennium several researchers proposed to move multicast functionality away from routers towards end systems [4], [5], [6], [7]. In these approaches, multicast-related features are implemented at end systems, assuming only unicast IP service. Moving multicast functionality to end systems has the potential to address many of the problems associated with IP multicast. Deployment is accelerated because all packets are transmitted as unicast packets. In addition, solutions for supporting higher layer features can be significantly simplified by using unicast solutions and application specific intelligence. But moving multicast functionality away from routers has obvious performance penalties. For example, preventing multiple overlay edges from traversing the same physical link is not completely possible so there will always be redundant traffic on physical links. Further, communication between end systems need the traversing of other end systems which

increases latency. Hence, many research efforts have focused on addressing these performance concerns.

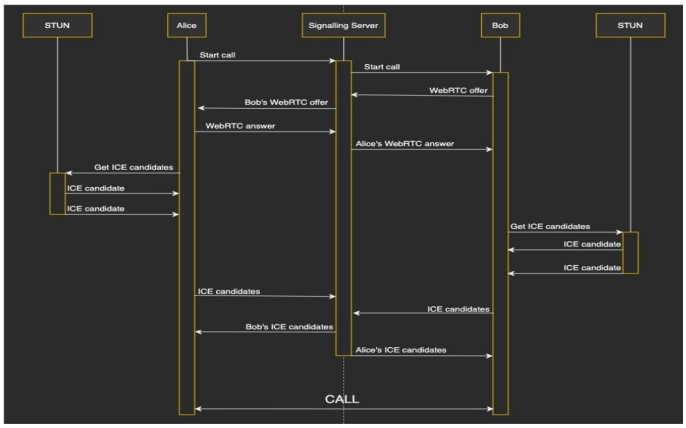
IV. WORKING



Because clients do not need to send and receive messages through a server, direct communication between browsers improves performance and minimises latency times. For example, we might connect two clients using WebSockets, but a server would have to transport their communications as shown in the picture below:
WebRTC, on the other hand, requires only a server to originate and control client connections.
Signaling is the term for this procedure. After the browsers have gathered the necessary information from their peers, they can communicate with one another as follows:
Because WebRTC does not specify a signalling messaging protocol, the implementation chosen must be depending on the application requirements; some of the most prevalent approaches include WebSockets, SIP, and others.

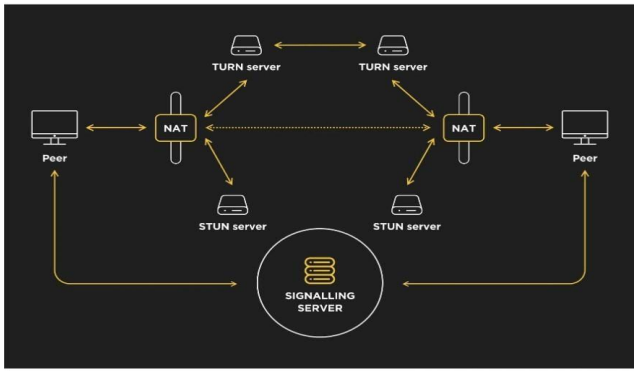
The following is how the signalling system works:

1) The call is started by a client.
2) The caller constructs an offer and delivers it to the other peer using the Session Description Protocol (SDP)(callee).
3) The callee answers with a response message that also includes an SDP description.
4) Both peers know the media capabilities used for the call once they've specified their local and remote session descriptions, which contain information like browser codecs and metadata. SDPs, on the other hand, will not be able to connect and exchange media data unless they learn about external NATs (Network Address Translators), IP addresses, and how to deal with any port restrictions. Interactive Connectivity Establishment is used to accomplish this (ICE).



What is Interactive Connectivity Establishment and how does it work? ICE is an Internet peer-to-peer network communication technique for sending and receiving media data. The details of routing (NATs, firewalls, etc.) are outside the scope of this article, but it is something that WebRTC must address. For NAT and firewall traversal, ICE aggregates available network connections, known as ICE candidates, and uses the protocols STUN (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT).
CE processes the connection in the following way:
It begins by attempting to connect peers directly through UDP.
If UDP fails, TCP is used. If both UDP and TCP direct connections fail, as they frequently do in real-world circumstances due to NATs and firewalls, ICE will connect peers via a STUN server with UDP. A STUN server is a server that uses the STUN protocol to determine a peer's public address and port behind an asymmetric NAT.

If the STUN server fails, ICE will fall back on a TURN server, which is a STUN server with additional relaying capabilities that can pass through symmetric NATs. As you can see, ICE tries to use STUN servers initially, but for severely restricted corporate networks, TURN relay servers will be required. TURN relay servers are expensive, so you'll either have to pay for your own or hire one, but ICE will usually be able to link peers with STUN. A STUN/TURN conversation is shown in the diagram below:



To summarise, the signalling mechanism is utilised to trade media data via SDP files, whereas ICE is used to communicate peer network connections. Peers can finally share data directly through their browsers after the channel has been established.

A. Features

1) Participant Count: Users may see how many people are currently in the meeting.
2) Administrator: The administrator is the person who created the room. He has the ability to delegate control to other users, mute other participants, and end the conference.
3) Controller: The controller is the user who has the ability to give a presentation or stream a local video.
4) Public Conversation: This is a general per-room chat where users can type a message that will be delivered to all members of the room of the space
5) Private Chat: A private chat takes place between any two participants in the room.
6) Videos Mode: Shows the video stream from all of the other cameras.
7) Presentation Mode: Enables the current controller to give a presentation that is accessible to all other users.
8) Video Feed Mode: In addition to his camera stream, the present controller can stream a locally stored video file to all other users.
9) Mute User: Allows the administrator to silence a specific user for the entire conference.
10) Toggle Audio/Video — Allows each user to choose whether or not to receive audio or video from a particular meeting participant.

B. Video Conferencing

When a user logs in, our main Audio/Video Conferencing module appears, allowing them to view the video of others and privately communicate with them. This module opens when a user clicks on a new module on the UI.

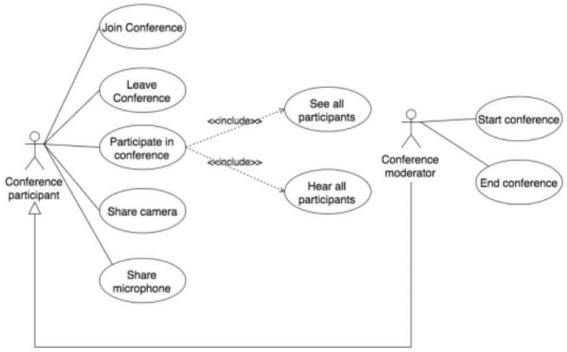


Fig 2: Video conferencing

Any video can be enlarged by simply clicking on it. Each video has a name tag, as well as additional admin controls.

C. Participant List and Features

A list of all linked participants appears on the right side of the screen. Next to each user are two buttons.

1) *Chat* - This button allows the user to have a private conversation with that individual.
2) *Download Audio/Video* - This button will download either the audio or video for that particular participant. This is an interactive and dynamic list. As the number of participants grows, the user can move through the list.

V. CONCLUSION

A video conferencing solution was designed and approved using WebRTC technology. WebRTC was chosen for this study because it is simple to use and does not require any plugins or applications to be installed; all you need is a browser that supports it. The video conferencing system runs on a web browser that runs on a range of operating systems. The purpose of this research is to reduce the amount of work and difficulties involved in communicating while also producing a video. a video conference that includes voice calls, video calls, file sharing, desktop sharing, and recording.

The format and attendance will vary depending on who attends. These goals have been achieved. The option to create and join rooms will be available to users. They can construct and join rooms, as well as broadcast and receive voice and video. They can receive audio and video broadcasts from other users and share presentations and locally recorded videos. They are also capable of communicating with one another. either using private or public texting Our app works on computers, tablets, and smartphones.

VI. OBJECTIVES OF PROPOSED

1) Main aim to create a Peer to peer conferencing.
2) Create a decentralized System for videoconferencing .
3) Introduce local file sharing in video conferencing.

VII. FUTURE SCOPE

Our application covers a wide range of topics, including business, distant learning, telecommuting/home offices, legal settings, and telemedicine. It allows businesses to meet and collaborate with others over long distances. Teachers and pupils can see one another. In an atmosphere comparable to a typical classroom, students can share documents and discuss issues. It assists users in by videoconferencing with clients and/or coworkers, you can save time and money. Videoconferencing technology is becoming increasingly widely used. It is becoming more widespread in today's courtrooms. The usage of videoconferencing technology allows testifying witnesses to appear in court without having to go to the courtroom. A patient in a remote area can consult with an expert from anywhere in the world.

VIII. RESULTS AND DISCUSSION

Following the completion of the WebRTC video conferencing system, the following results were obtained:

1) A video conferencing solution for the web was created.
2) Users can create and join rooms.
3) Voice and video calls are available for users.
4) Users in the same communication room can communicate with one another.
5) The user who initiates the chat session has the ability to Know how to answer the phone
6) Using this, the system reduces physical effort Instead of a live meeting a video conference technology is used
7) It allows for simple and straightforward communication.
8) The user can attend or make a call from any location or at any time.

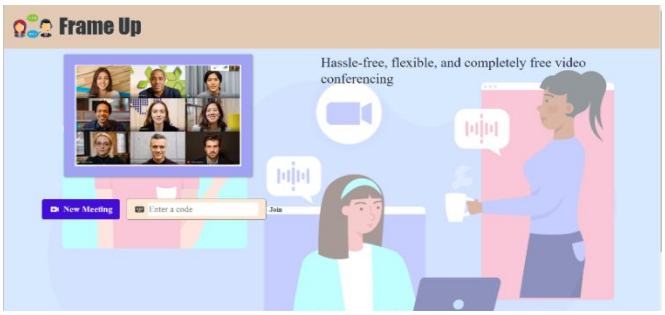


Fig 3 UI Interface

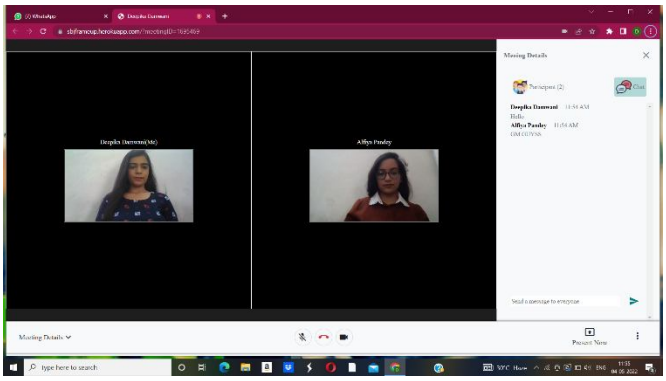


Fig 4 Video Conferencing Window

REFERENCES

[1] Rhinow, Florian, Pablo Porto Veloso, Carlos Puyelo, Stephen Barrett, and Eamonn O. Nuallain. "P2P live video streaming in WebRTC." In Computer Applications and Information Systems (WCCAIS), 2014 World Congress on, pp. 1-6. IEEE, 2014.

[2] Liu, Jiangchuan, Sanjay G. Rao, Bo Li, and Hui Zhang. "Opportunities and challenges of peer-to-peer internet video broadcast." Proceedings of the IEEE 96, no. 1 (2008): 11-24.

[3] Deering, Stephen E., and David R. Cheriton. "Multicast routing in datagram internetworks and extended LANs." ACM Transactions on Computer Systems (TOCS) 8, no. 2 (1990): 85-110.

[4] Chawathe, Yatin, Steven McCanne, and Eric Brewer. "An architecture for internet content distribution as an infrastructure service." Unpublished, available at http://www.cs.berkeley.edu/yatin/papers (2000).

[5] Chu, Yang-hua, Sanjay G. Rao, Srinivasan Zeeshan, and Hui Zhang. "A case for end system multicast." IEEE Journal on selected areas in communications 20, no. 8 (2002): 1456-1471.

[6] Francis, Paul. "Yoid: Extending the internet multicast architecture." (2000).

[7] Zhuang, Shelley Q., Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination." In Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video, pp. 11-20. ACM, 2001.

[8] Chu, Yang, Sanjay Rao, Srinivasan Seshan, and Hui Zhang. "Enabling conferencing applications on the internet using an overlay muilticast architecture." ACM SIGCOMM computer communication review 31, no. 4 (2001): 55-67.

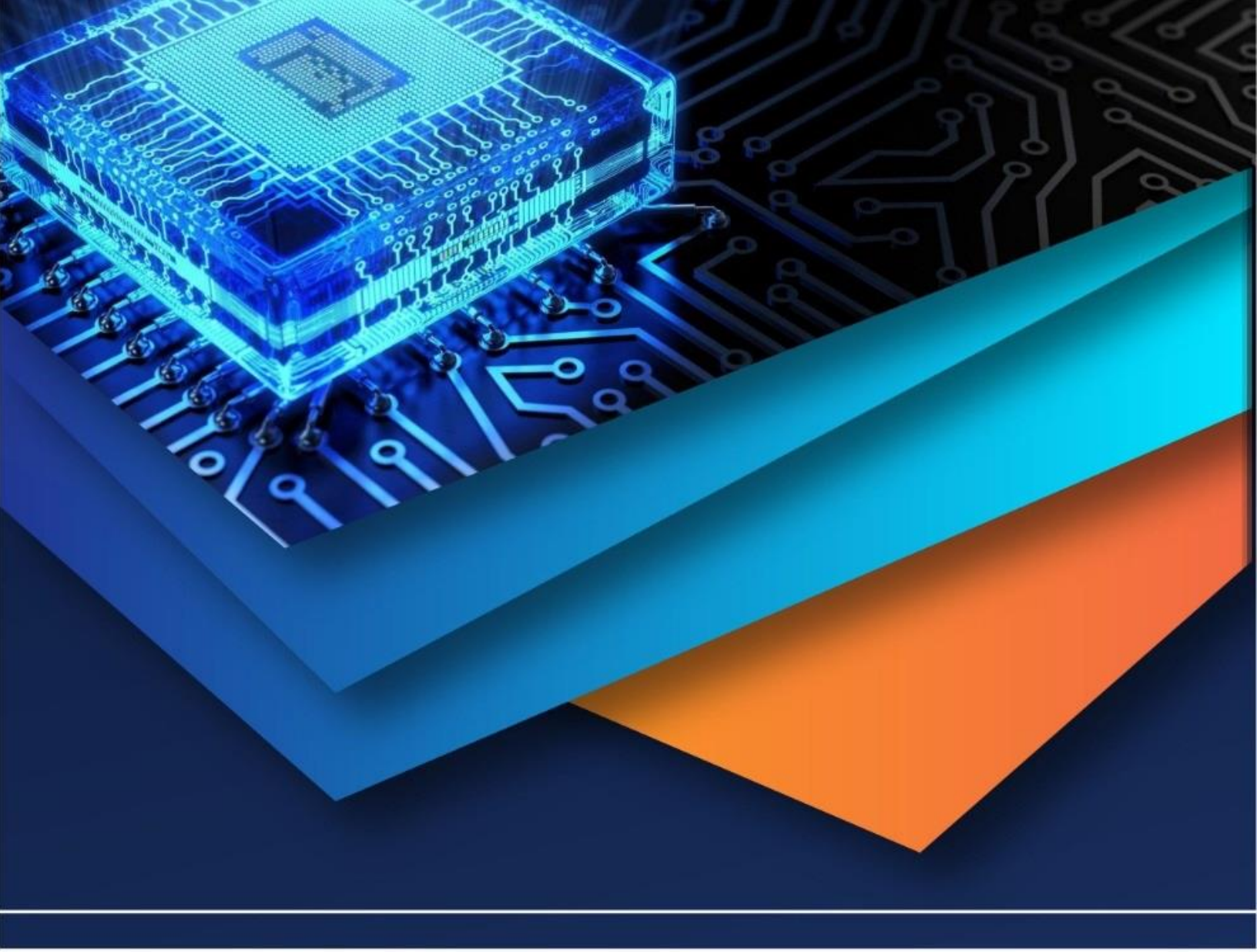
[9] Luo, Chong, Wei Wang, Jian Tang, Jun Sun, and Jiang Li. "A multiparty videoconferencing system over an application-level multicast protocol." IEEE Transactions on Multimedia 9, no. 8 (2007): 1621-1632.

[10] Chen, Minghua, Miroslav Ponec, Sudipta Sengupta, Jin Li, and Philip A. Chou. "Utility maximization in peer□to-peer systems." In ACM SIGMETRICS Performance Evaluation Review, vol. 36, no. 1, pp. 169-180. ACM, 2008.

[11] Ponec, Miroslav, Sudipta Sengupta, Minghua Chen, Jin Li, and Philip A. Chou. "Multi-rate peer-to-peer video conferencing: A distributed approach using scalable coding." In 2009 IEEE International Conference on Multimedia and Expo, pp. 1406-1413. IEEE, 2009.

[12] Koh, Eunyee. "Conferencing room for telepresence with remote participants." In Proceedings of the 16th ACM international conference on Supporting group work, pp. 309-310. ACM, 2010.

[13] Isaacs, Ellen A., and John C. Tang. "What video can and cannot do for collaboration: a case study." Multimedia Systems 2, no. 2 (1994): 63-73.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY