International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 02 | Feb 2022 www.irjet.net p-ISSN: 2395-0072

A Decentralized Application for Secure Private and Group Messaging in a Peer-to-Peer Environment

Badgujar Niranjan Sanjay[1], Nair Aditya Sivaram[2], Nair Sidharth Shankaranarayanan[3], Vishwakarma Anant Jaiprakash[4] and Dr. Sharvari Govilkar[5]

[1,2,3,4] *UG Student, Dept. of Computer Engineering, Pillai College of Engineering, New Panvel, Maharashtra, India*
[5] *Head of the Dept., Dept. of Computer Engineering, Pillai College of Engineering, New Panvel, Maharashtra, India*

Abstract - *Text messaging in a peer-to-peer environment is a concept that has been explored in the past using various technologies available at that time. The peer-to-peer model does not use a central database or computation server, thus eliminating single points of failure and increasing the availability and reliability of the network. It also provides opportunities to improve privacy and confidentiality since messages are not being routed through a server but are directly sent to the recipient. Previous work has focused on private one to one messaging between peers. Also, both peers needed to be online at the same time in order for the text message to be sent to the recipient. The proposed solution uses WebRTC, a modern peer to peer web communication technology as the underlying framework that facilitates secure communication. A variation of the Proof of Authority algorithm which is normally used in blockchain technology will be used to facilitate secure and reliable group messaging. A provisional node management system will be established so that messages can be transmitted across the network even if the recipient is offline at the time of transmission. The message transmission delay will be analyzed to ensure optimum performance. The queue size for unsent messages will be optimized so that it does not consume excess memory and storage. The provisional node management system will be tested to include hard limits such that bandwidth used is in the acceptable range.*

Key Words*: *Decentralized Application (dApp), Peer-to-Peer (P2P) Networks, WebRTC, Blockchain

1. INTRODUCTION

Traditional chat applications are centralized i.e., all the data is stored or passed through a centralized server. Therefore, the major problem of this structure is, if the central server fails then the whole network collapses. For example, WhatsApp stores all the data on a central server, if in case that server is destroyed then there can be a loss of user data, or they can even leak the user information stored on the server. To overcome this, Decentralized Applications (dApps) are useful as the data is not stored on the central server and it also uses encryption to provide security. Due to these features, dApps are gaining popularity and are replacing centralized systems gradually.

2. LITERATURE SURVEY

A. WebRTC: Shikhar Vashishth et al. [18] proposed a peer to peer (p2p) architecture to exchange messages asynchronously and facilitate new peer joins via existing peers in the network, thus reducing the dependency on the bootstrap server. The data channel of WebRTC was used to implement the browser compatible framework. It is a modified version of Chord (a distributed lookup protocol for p2p networks). Multiple STUN servers were used to balance load of ICE candidate requests on them. A TURN server was used to route traffic via the relay server. Further, periodically existing connections were checked for whether or not they were required and if not, destroyed.

B. Etherium: Sourabh et al. [4] proposed an implementation of a decentralized chat application on the Ethereum blockchain network with ReactJS. Infura was also used which is a hosted Ethereum node cluster that lets users run the application without requiring to set up their own Ethereum node or wallet. Solidity was used which is an object-oriented, high-level language for implementing smart contracts (programs which govern the behavior of accounts within the Ethereum state). Also used Etherscan which allows to explore and search the Ethereum blockchain for transactions, addresses, tokens, prices and other activities. The AES-256 algorithm was used for encryption and decryption using CryptoJS.

C. Interplanetary File System (IPFS): Faraz Khan et al. [5] proposed a Dchat that uses IPFS technology for decentralized anonymous and tamper-proof cross platform communication systems with user friendly interface. IPFS establishes a permanent distributed network by using content-based reference. It uses the hash of the file itself to be used as a reference. It also gives the users an advantage to be anonymous during the communication. The system uses IPFS hashes for security which are 32 bytes with SHA256 algorithm. LibP2P protocol was used to listen for incoming messages and

broadcasts and directly messaging to peers on the network.

2.1 SUMMARY OF RELATED WORK

The summary of methods used in literature is given in Table 1.

Table - 1: Summary of literature survey

Literature	WebRTC	Etherium	IPFS
Shikhar Vashishth et al. 2016 [18]	Yes	No	No
Sourabh et al. 2020 [4]	No	Yes	No
Faraz Khan et al. 2020 [5]	No	No	Yes

3. PROPOSED WORK

The proposed system uses different technologies like WebRTC, Blockchain and various other concepts like encryption, consistency, availability and techniques to improve network stability to create a robust peer to peer decentralized application.

3.1 SYSTEM ARCHITECTURE

The system architecture is given in Figure 1. Each block is described in this Section.

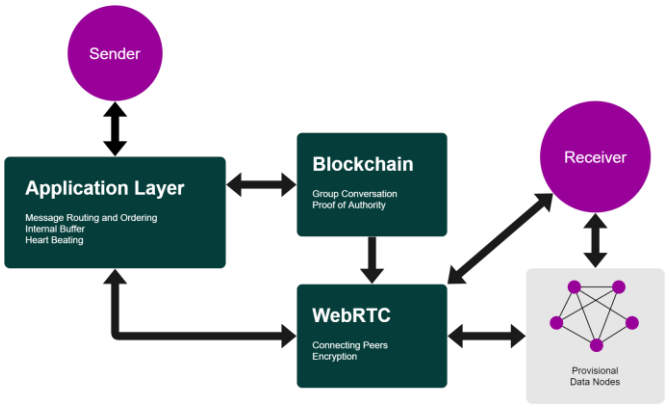

Fig - 1: Proposed system architecture

A. WebRTC module: The WebRTC module is the foundation of the entire architecture. It is responsible for connecting two peers over the internet as well as providing end to end encryption using HTTP over TLS (HTTPS) for any communication occurring between them. However, issues arise due to the limited address space of the IPv4 protocol. Each client has their own private IP address as well as a public IP address. WebRTC can only connect two peers if their public IPs are known. To solve this problem, STUN and TURN servers are used. A STUN server only comes into play during the connection phase. Once the two peers are connected, all data is transmitted directly between them. However, in some cases this is not possible e.g., when one of the peer's network is masqueraded by a symmetric NAT. A TURN server is used in this case to act as a proxy between the two peers. Since all data transmitted has to pass through the TURN server, it results in bandwidth costs for the TURN server. Hence, TURN servers are not freely available. Thus, the architecture requires us to implement a few TURN servers of our own. Although this makes our system seem like a more centralized architecture due to the reliance on TURN servers, when looked through the application's point of view, the TURN server is only responsible for connecting peers and facilitating communication between them. These servers do not store any application data, thus if one TURN server is down, it does not affect the application performance as long as another TURN server is running (if network latency is ignored and server is considered to be ideal). Thus, it can be concluded that the system architecture is still decentralized.

B. Client / Application layer: There are various issues that can arise when a peer tries to send messages to another peer. The data may be received out of order, or corrupted, or lost. To deal with these issues certain mechanisms have to be implemented in the application. An internal message queue is used as a buffer to store messages that have yet to be successfully sent. For a message to be successfully sent, the sender must have an active internet connection, and the receiver must reply with an acknowledgement before a specified timeout expires. Once this happens, the message is removed from the queue. A message digest created using the SHA-256 algorithm is included in the sent data that can be used by the receiver to verify the integrity of the received message. A nonce is also included in the message, which ensures that the received message is not processed more than once by the receiver. A serial message identifier is also included in the message to ensure that the messages are not processed out of order. Explicit encryption does not have to be implemented unless necessary (for e.g., when using the provisionary data node layer) because end to end encryption is already provided by the WebRTC module. Heart Beating is done with the receivers to check if they are online in order to send them messages.

C. Provisionary data node layer: The WebRTC connections can only work when both the peers are online at the time of data transmission. However, in a messaging system it is often the case that the receiver is offline at the time of message transmission. Although this can be solved with a message queue, an issue arises when the receiver comes back online but the sender is currently offline. Now the sender cannot send the message due to lack of internet connectivity. To solve this problem, the message can be sent to one or more intermediate or provisionary data nodes along with the recipient's address. These data nodes can be picked from the sender's contact list. The provisionary nodes would heartbeat the recipient to check their online status and forward the message to them once they become online. The nonce facility from the application layer can be used to deal with duplicate messages received from multiple such nodes. The RSA asymmetric encryption algorithm is used to encrypt messages sent to provisionary data nodes. For end-to-end encryption to be maintained, the public keys should already be exchanged between the sender and receiver through the WebRTC module. This facility raises minor privacy concerns however, as the sender and recipient addresses become known to a third party. Although they cannot decrypt the message that is being sent, they can come to know who is communicating to what, and when they are communicating.

D. Blockchain module: Although the above modules are sufficient to facilitate communication, it becomes increasingly inefficient in the case of group messages. One sender would have to individually send messages to all group members, and maintain individual internal message queues for them and wait for their acknowledgements. This would further lead to congestion in the provisionary data node layer as one message is being duplicated multiple times. To solve this problem, private blockchains can be used. Every group would have their own private blockchain that would store records of messages in it's blocks. The individual messages are digitally signed (using RSA) by the sender's private key for verification. The messages are encrypted by a symmetric shared secret key using the AES-256 algorithm of which every group member has a copy. Proof of Authority is used as the consensus mechanism where the authority belongs to the group admins. Each block inside the blockchain is digitally signed (using RSA) using the group admin's private key for verification. So in case a user comes online after a long time and wants the new blocks in the blockchain, they can ask the other online users in the group for the blocks even if an admin is not currently online and securely accept all blocks that have been digitally signed by any group admin. The system works better if a group has more than one admin. A group member who wants to send a message to the group would send it to the online group admins. Once an admin receives such a message, they add it to the current block that they are creating. Once a specified time interval has passed (3 to 5 seconds), no more messages will be added to that block. The admin is going to digitally sign the block using RSA and append it to his blockchain copy. The admin transmits the new block to all the group members. A time based nonce is used so that duplicate blocks received from other online admins do not cause problems. The drawback with this approach is that if an admin is not online, group messages are no longer able to be sent by any member.

4. REQUIREMENT ANALYSIS

The implementation detail is given in this section.

4.1 ALGORITHMS

Encryption: TLS over HTTP (HTTPS) uses RSA encryption, RSA encryption for sending messages to provisionary data nodes, AES-256 encryption for shared symmetric key encryption.

Hashing: SHA-256 (for message digest & verifiable tokens)

Encoding: Base64 encoding for invite link generation.

Digital Signatures: RSA

4.2 HARDWARE

The hardware specifications are given in Table 2.

Table - 2: Hardware specifications

Processor	2 GHz Intel
HDD (Free Space)	2 GB
RAM	8 GB

4.3 SOFTWARE

The software specifications are given in Table 3.

Table - 3: Software specifications

Operating System	Windows 10 or later
Programming Language	HTML, CSS, JavaScript, JSX

5. CONCLUSION

Peer-to-peer messaging is more secure and reliable than a messaging application with a centralized server. Decentralized environments are a better option than a centralized one since the whole system will shutdown in case the central server goes down. Group messaging can be made efficient using blockchain concepts.

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our supervisor and Head of the Department Dr. Sharvari Govilkar for the valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of this work. We deeply express our sincere thanks to our Principal Dr. Sandeep M. Joshi for encouraging and allowing us to present this work.

REFERENCES

[1] Vikas Bhardwaj, Raja Ram Sharma, Akhilesh Kumar, "Chat Application using Blockchain Technology", International Journal for Research in Engineering Application & Management (IJREAM), 04 Apr 2021.

[2] George Suciu, Stefan Stefanescu, Cristian Beceanu, Marian Ceaparu, "WebRTC role in real-time communication and video conferencing", Global Internet of Things Summit (GIoTS), 2020.

[3] Christos Aslanoglou, Michalis Konstantopoulos, Nikos Chondros, Mema Roussopoulos, "Take Back your Friends with DCS: A Decentralized Connectivity Service for private social communication apps", IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), 2020.

[4] Sourabh, Deepanker Rawat, Karan Kapkoti, Sourabh Aggarwal, Anshul Khanna, "bChat: A Decentralized Chat Application", International Research Journal of Engineering and Technology (IRJET), May 2020.

[5] Faraz Khan, Niraj Mantri, Sagar Rajput, Dhananjay Dhakane, Puja Padiya, "Anonymous De-centralized Ephemeral Chat Application using Interplanetary File System", ITM Web of Conferences 32, 02004, 2020.

[6] Kaidong Wu, Yun Ma, Gang Huang, Xuanzhe Liu, "A First Look at Blockchain-based Decentralized Applications", Key Lab of High-Confidence Software Technology, 3 Sept 2019.

[7] Kahina Khacef, Guy Pujolle, "Secure Peer-to-Peer communication based on Blockchain", 33rd International Conference on Advanced Information Networking and Applications (AINA-2019), Matsue, Japan, Mar 2019

[8] Mohamed Abdulaziz, Davut Çulha, Ali Yazici, "A Decentralized Application for Secure Messaging in a Trustless Environment", International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism, Ankara, Turkey, 3-4 Dec 2018.

[9] Nileshkumar Pandey, Doina Bein, "Web Application for Social Networking using RTC", IEEE Annual Computing and Communication Workshop and Conference (CCWC), 2018.

[10] Wei Cai1, Zehua Wang, Jason B. Ernst, Zhen Hong, Chen Feng, Victor C.M. Leung, "Decentralized Applications: The Blockchain-Empowered Software System", Natural Sciences and Engineering Research Council of Canada, 2018.

[11] Abhishek P. Takale, Chaitanya V. Vaidya, Suresh S. Kolekar, "Decentralized Chat Application using Blockchain Technology", International Journal for Research in Engineering Application & Management (IJREAM), 2018.

[12] Yunhua He, Hong Li, Xiuzhen Cheng, Yan Liu, Chao Yang, Limin Sun, "A Blockchain based Truthful Incentive Mechanism for Distributed P2P Applications", Natural Science Foundation of Shaanxi Province, 2018.

[13] Hiroyoshi Ichikawa, Aki Kobayashi, "Collaborative Messaging Protocol with Multiple Intermediate Nodes", 7th International Congress on Advanced Applied Informatics, 2018.

[14] Nikita V. Ghodpage, Prof. R. V. Mante, "Privacy Preserving and Information Sharing in Decentralized Online Social Network", 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018).

[15] Peter Menegay, Jason Salyers, Griffin College, "Secure Communications Using Blockchain Technology", Milcom 2018 Track 3 - Cyber Security and Trusted Computing, 2018.

[16] Yongle Chen, Hui Li, Kejiao Li, Jiyang Zhang, "An improved P2P File System Scheme based on IPFS and Blockchain", IEEE International Conference on Big Data, 2017.

[17] Paulo Chainho, Steffen Drusedow, Ricardo Lopes Pereira, Ricardo Chaves, Nuno Santos, Kay Haensge, Anton Roman Portabales, "Decentralized Communications: Trustworthy Interoperability in Peer-To-Peer Networks", 2017.

[18] Shikhar Vashishth, Yash Sinha, K Hari Babu, "Addressing Challenges in Browser Based P2P Content Sharing Framework Using WebRTC", IEEE 30th International Conference on Advanced Information Networking and Applications, 2016.

[19] Cristian Constantin Spoiala, Alin Calinciuc, Corneliu Octavian Turcu, Constantin Filote, "Performance comparison of a WebRTC server on Docker versus Virtual Machine", 13th International Conference on Development And Application Systems, Suceava, Romania, May 19-21, 2016.

BIOGRAPHIES

Badgujar Niranjan Sanjay

Nair Aditya Sivaram

Nair Sidharth Shankaranarayanan

Vishwakarma Anant Jaiprakash

Dr. Sharvari Govilkar
She is a professor and Head of Computer Engineering Dept. at PCE, New Panvel. She has more than 24 years of teaching experience in the field of Computer Engineering. She has published about 81 research papers in various national and international conferences and journals of repute with google citation count as 503.