Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 173 (2020) 158–170

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on *Smart Sustainable Intelligent Computing and Applications* under ICITETM2020

Decentralized and Secure Communication Architecture for FANETs using Blockchain

Kashish Khullar[a], Yishu Malhotra[b], Anupam Kumar[c,*]

[a]*Newgen Software, Tower E, Sector 144, Noida, Uttar Pradesh 201301, India*
[b]*University of Montréal, 2900 Edouard Montpetit Blvd, Montreal, Quebec H3T 1J4, Canada*
[c]*Department of Computer Science, Maharaja Agrasen Institute of Technology Rohini, Sector 22, PSP Area, Delhi, 110086, India*

Abstract

Flying Ad Hoc networks, (FANETs) in recent years, have actively been used in monitoring landscape, military and mapping terrains. However, with the advent of the emerging concept of smart cities and new business applications, these flying devices and drones have found a new use case in the medical and governance sector. But these networks suffer from the major problem of centralization. Although centralization does provide reliability and efficiency but also poses the threat of a single point of failure. Meanwhile, blockchain widely known for its decentralization is heavily researched and can be utilized in this case to offer a solution to the problem of centralization of FANETs. Therefore, in this paper, we propose a decentralized architecture of flying ad hoc nodes based on blockchain and using Practical byzantine fault tolerance (PBFT) for consensus among nodes. By employing PBFT, the architecture is not only computationally efficient and fast. In addition, we have used a gossip protocol for passing messages among nodes. Lastly, we have simulated the working of our model and the experimental results show that the proposed method works with nearly constant throughput and latency while increasing the network size and approximately constant message overhead with increase transaction for given network size.

Keywords: Flying ad hoc Networks; Communication Architecture; Vehicle to vehicle; Blockchain; Byzantine fault; Security and Privacy;

* Corresponding author.
E-mail address: kashish.khullar@newgen.co.in (Kashish Khullar), yishu.malhotra@umontreal.ca (Yishu Malhotra), anupamkumar@mait.ac.in (Anupam Kumar)

1877-0509
10.1016/j.procs.2020.06.020

1. Introduction

Internet of things is one of the most disruptive technologies of this century. [1] In recent years, businesses have increased the use of IoT devices to provide multiple services to people ranging from communication to the development of smart cities. Generally, each organization owns a set of IoT devices that are used for providing services, however, there has been in shift in the architectural paradigm in the internet of things space, where once we had a central server which monitors the entire working of the network, we are gradually adopting decentralized and distributed ad hoc networks for their systems.

With the perpetual growth and adoption in the field of IoT, we have witnessed a rise in the application of Mobile Ad Hoc Networks (MANETs), Vehicular Ad Hoc Networks (VANETs) and Flying Ad Hoc Networks. MANETs is an area in the distributed computing discipline that has been heavily researched, they are a network of mobile nodes that are connected wirelessly and undergo frequent network topology changes due to which they don't have a fixed infrastructure. [2] Emerging from MANETs is another class of ad hoc networks known as FANETs that have potential use cases in a smart city environment. FANETs have previously been used before in military, surveillance of an area, film making and emergency situations during disasters. FANETs have increased scalability and sustainability as even if one of the UAVs in the network fails, other UAVs can continue to perform the given task. Due the mobility of nodes in FANETs, there is no fixed topology implemented in the network. [2]

Furthermore, both these classes of networks suffer from the problem of centralization. It is not uncommon for a central authority such an institution or government agency to control such networks [3], rendering them as a single point of failure which can be exploited by attackers outside the system. Another problem that such networks do not inherently support is public infrastructure for voluntary inclusion of nodes in the system. Without any incentive or compensation, joining and contributing to a network would be infeasible for the public nodes. Centralization of networks, not only creates a security concern but also requires each node to trust a central node in the system.

Blockchain can serve as a solution to this problem. Since its invention in 2008 [4], blockchain has been disrupting many industries by extending its concepts in multiple use cases. Being a decentralized data structure powered by a consensus algorithm, distributed nodes, and cryptography, blockchain can assist in creating a decentralized network of nodes without any central authority. In such a decentralized environment, blockchain removes the need to trust every node in the network. The consensus algorithm employed by the blockchain ensures that the state of the ledger is consistent across all nodes in the network.

In this paper, we propose a concept to use blockchain technology in FANETs using the byzantine fault-tolerant consensus algorithm and gossip protocol for communication. This architecture allows multiple organizations to collaborate their nodes to form a large network of nodes that can be used to provide services to the public over an increased area coverage and blockchain, in this case, offers security and decentralizes the network by removing a central authority. The main contributions of this paper are as follows:

- Implement the working of gossip protocol in ad hoc networks.
- Illustrate the working of Practical Byzantine fault tolerance algorithm.
- Implement blockchain with respect to flying ad hoc networks.
- Propose a decentralized architecture for FANETs.

The remainder of this paper is organized as follows: Section II briefly describes FANETs, followed by the working of blockchain technology, practical byzantine fault tolerance and gossip protocol. Section III reviews the related works. Our proposed architecture and algorithms used are illustrated in Section IV. Experimental Setup and result analysis for this paper are provided in Section V and VI respectively. In Section VII, we describe a few possible applications of the proposed architecture. We finally conclude this paper in Section VIII and provide future scope in Section IX.

2. Background

2.1. Flying Ad Hoc Networks

Technological advancements in electronic and communication systems have catalyzed the use of small Unmanned Aerial Vehicles (UAVs) in various sectors such as commercial, military, etc. FANET may be defined as an upgraded form of MANET in which UAVs act as nodes of the network. It is a group of UAVs that fly without much human intervention or carrying some human personnel, like autopilot.

FANETs belong to a subcategory of VANETs which is a sub form of MANETs, hence FANETs can be formed only by using several UAVs but not the single UAVs. But in order to call the multi-UAVs system FANETs, it needs to ensure that communication among nodes is being realized using the ad hoc network that means UAVs must not rely solely on the infrastructure-based links.

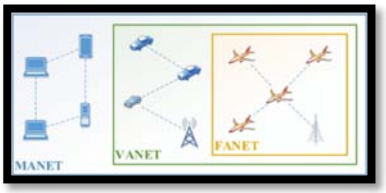

Fig. 1. Comparison between MANET, VANET and FANET. [5]

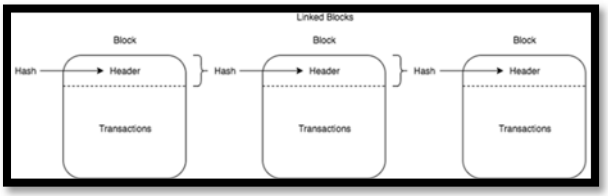

Fig. 2. List of blocks in a Blockchain [8]

2.2. Blockchain Technology

Blockchain is a decentralized ledger, shared among multiple nodes in a network which cannot be trusted. Blockchain is a tamper-proof record of transactions [6] or events that change the state of the blockchain when executed. In a blockchain, blocks contain the transactions that have been created in a single round. The tamper-proof property of blockchain is attributed to the links between each block in the chain. Every block except the genesis block is linked to the previous block via a hash pointer [7], which is the hash value of the previous block, shown in figure [8]. As a result, the content of the next block directly depends on the content of the current block. Any tampering in any block of the chain changes the hash value of the block next to it. We can, therefore, verify the data of the entire chain since the beginning.

Blockchain networks such as Bitcoin [4] [9][10] and Ethereum [11] use consensus algorithms called proof of work which determines which node will get the ability to add a new block to the ledger. Proof of work requires nodes or miners to compute the hash of a block until a given condition is fulfilled. The fastest miner or node to do so adds the block to the chain and broadcasts its solution to the network.

2.3. Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance [12] is a consensus algorithm used in distributed systems that solves the Byzantine general problem through distributed voting. Theoretically, the PBFT algorithm can tolerate f malicious node in a system given 3f+1 node. Hyperledger fabric and Tendermint [13] both use a modified version of PBFT for consensus in their blockchain. Modified PBFT algorithm in Tendermint works by sending messages to its peers in the network using gossip protocol. Messages sent across the network include PROPOSAL, PREVOTE, PRECOMMIT messages. The proposer proposes a value broadcasts a PROPOSAL message to the rest of the nodes in the systems. Each node on receiving the particular PROPOSAL for a round r, the node sends a PREVOTE message to confirm a value. It later waits for a predetermined time period until it receives 2f+1 PREVOTES on a value that represents the acceptance of the value v by the network. This initiates the last phase in which the node locks the value v for the round and sends the PRECOMMIT message. When a node receives 2f+1 PRECOMMIT messages of round r and value v, the node commits the value v, reflecting the change in the state of the system.

2.4. Gossip Protocol

Gossip Protocol is a communication protocol in computers that is inspired by the way epidemics spread. Distributed systems use this protocol for sending messages to all the nodes in the network in a short span of time. Particularly in ad hoc networks gossip protocols are effective since they do not a well-defined structure. Gossip protocol works by selecting a set of neighbours and sending a message to neighbouring UAV which further does the same their neighbours and so on until the entire network has received the same message.

Gossip protocol varies from flooding algorithms since it employs a probabilistic strategy [14] before sending the messages further to its neighbours. Unlike flooding algorithms, where messages are broadcasted as soon as they are received, nodes broadcast the message further based on a condition. [15] Tendermint also uses this protocol for broadcasting messages to the network. Being fast and continuous gossip protocol allows a node to synced up quickly even if a node is delayed or disconnect for a long duration.

3. Literature Review

Blockchain technology has found many applications in the field of IoT. Many papers have been published in their respect in the past.

Dorri et. al have proposed architecture in [16] for IoT which uses blockchain in smart home devices. Their proposed architecture works by recording sensor data and creating a transaction with the data as a payload and transmitting it to the other clustered nodes and cloud storage which holds a private blockchain. Since this architecture works on a private blockchain present in cloud storage, the evaluated it to resilient to many threats.

Christidis in [17] has reviewed the working IoT devices and how smart contracts can be used in the IoT sector. It also mentions how blockchain enables the creation of market place and resource sharing in the IoT sector and automates the existing infrastructure using cryptography.

There have been multiple protocols published in favor of blockchain and IoT [18] and [19] provide a comprehensive survey of the protocols and applications of blockchain in the IoT.

Blockchain has also been used in MANETs. MONET [20] is an open-source project based on mobile ad hoc networks that allow inter blockchain communication. They have developed their own byzantine fault-tolerant consensus algorithm called Babble, which unlike a conventional blockchain is a Directed Acyclic Graph and FIP protocol for communication. Since MANETs are decentralized, Sybil attacks can be leveraged to exploit the networks. Unchained [21] is a blockchain-based solution that addresses this problem by managing identities and authenticate nodes using blockchain.

VANETs have been extensively been researched and many papers have been published in its respect. Ad hoc networks, being utilized in an open environment are vulnerable to multiple known and unknown attacks. Han et al. [22] , proposed an intrusion detection system for vehicular networks that detects unknown attacks and performs real-time analysis on vehicle data for survival estimation. Sharma and Kaul [23] have presented an extensive survey of various attacks possible on VANETs and VANET Cloud. Further, they elaborated on the concept of IDS and analyzed multiple IDSs and compared each technique the merits and demerits. To detect the zero-day attack, they have also proposed a Honeypot optimized IDS which achieves this with minimal overhead.

Ad hoc networks do not have a fixed structure since each node is mobile and can move with celerity. Thus routing protocols used for packet delivery must be reliable and efficient which can tolerate frequent disconnections. Hanshi et al. [24] reviewed various routing algorithms used in VANETs. Their paper focuses majorly on forwarding strategies and evaluates the applicability of existing protocols. Moreover, it also considers various factors such as channel modeling, vehicle density and geographic information that influence the working of such protocols in VANETs. Furthermore, other parameters, specific to the vehicle such as radio transmission range, relative velocity, and vehicle traffic are also considered while discussing the routing issues in VANETs. Boussoufa-Lahlah et al [25] presented a paper surveying merits and demerits of various geographic routing protocols and explores the design challenges of such protocols

Yang et al. [26] proposed a mechanism to enhance security in VANETs by using a privacy-focused authentication scheme. The proposed scheme employs a modular square root method to improve security. It consists of several phases, including system initialization phase where a vehicle's identity is registered and a certificate is

assigned, identity verification phase, communication phase in which digital signature of the message is computed and shared and finally a tracing, revocation and update phase where a malicious vehicle can be tracked using its id and prevent the further use of its certificate by adding the vehicles information in a revocation list. Afterward, the relevant vehicle list is updated correspondingly.

Researchers previously have used VANETs with blockchains. [27] proposes an architecture in which vehicular nodes communicate with each other and also to a decentralized application deployed on the Ethereum blockchain. The deployed application allows ease traffic management, enforcing rules and imposing vehicular taxes.

Singh et. al, [28] have proposed a trust-based framework for intelligent vehicular nodes to communicate with each other and share data without interference of other vehicle nodes by using blockchain technology. This system uses a consensus algorithm proof of driving which verifies the validity of the vehicle in the network. Sharma et al [29] have proposed a secure distributed transport management system based on blockchain and vehicular ad hoc networks. Their architecture allows the vehicle to communicate which each other and share resources.

4. Proposed Method

In this section, we elaborate on the components and working of the simulation of our concept. The following figure illustrates the entire working of the systems.

4.1. Flying Nodes

FANETs architecture consists of flying nodes that may or may not have a predefined structure and constantly change location while moving through their trajectory. Each node in our system majorly consists of a local blockchain, transaction pool, wallet, p2p-server, and routing table.

4.2. Mobility Management

Perpetual movement of nodes in our system can cause nodes to move out of the transmission range of the nodes and hence impede message transmission and reception. Hence, for multiple nodes to leave or join the network we need a peer discovery algorithm. Therefore, we need a dynamic peer discovery algorithm so that nodes can disconnect to nodes that fly away far from the current node and connect to nearby nodes.
To implement this functionality, we have used a shared routing table that is constantly updated with the latest location of the corresponding node. The following procedure shows the working of disconnect, connect and reconnect.

Algorithm 1 – Fanet dynamic neighbour generation algorithm

```
nodes ≔ getNodes()
neighbours[] ← nil
txRng ≔ getTransmissionRange()
Function updateLocation(node):
        if location(node) = nextLocation(node) then
                generateNextLocation(node)
        else
                X ≔ nextLocation(node)[0] – location(node)[0]
                Y ≔ nextLocation(node)[1] – location(node)[1]
                distance ≔ euclideanDistance(location(node),
                                                nextLocation(node))
                if distance > speed(node) then
                        location(node) ≔ [(location(node)[0] + X *
                        speed(node)/distance),(location(node)[1] + Y * speed(node) / Distance)]
                else
                        location(node) = nextLocation(node)
                        broadcast <LOCATION,round,nextLocation(node)>
                        generateNeighbours(node)

Function generateNextLocation(node):
        nextLocation(node) = [randrange(length(node),
```

```
                                        randrange(width(node))]

Function generateNeighbours(node,nodes,txRng):
        neighbours ← nil
        for adjNode in nodes:
                if node != neigbour ∧  distance(location(node),
                location(adjNode)) ≤ txRng
                        push(neighbours, adjNode)
        reconnect(node,neighbours)

Function reconnect(node, neighbours):
        for neighbour in neighbours:
                if connectionState(neighbour)!= READY_STATE then
                        connect(neighbour)
```

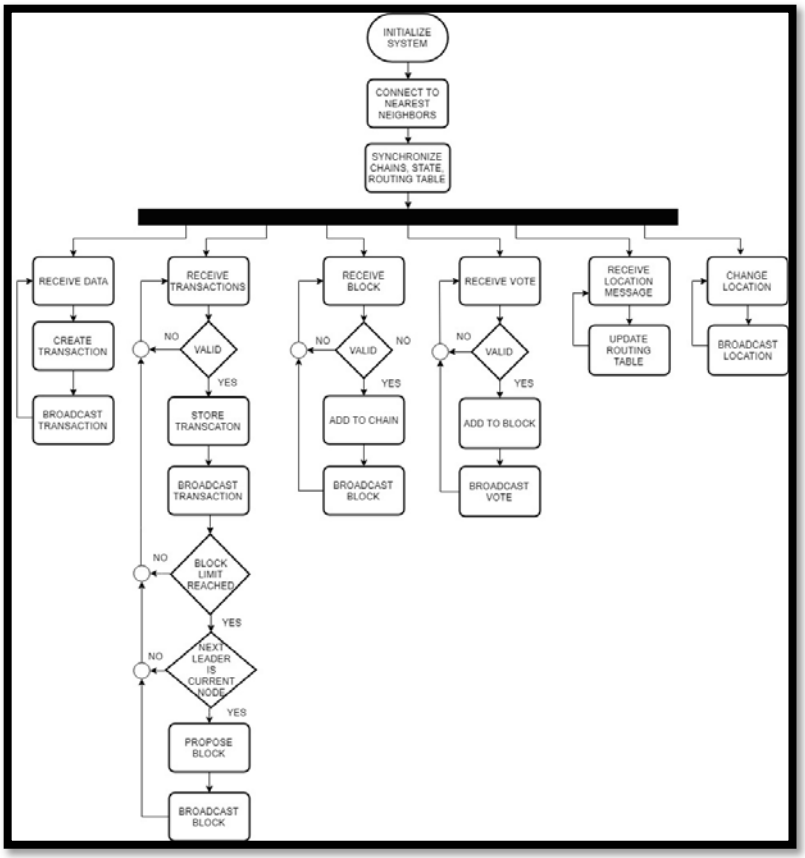

Fig. 3. Proposed Model

This algorithm is designed to dynamically detect the neighbours of a node. It first generates a set of nodes with the same transmission range for all the nodes. The updateloacation function updates the location of each node after executing a set of conditions. The generatenloaction function is called for each node in order to generate their next location randomly. The next location generated for each node is a random location in order to hold on to the basic definition of FANET nodes. The generateNeighbours function updates the list of neighbours for each node. If a node is within the transmission range of another node, then it will be called its neighbour.

4.3. Inter-node communication

Nodes in the proposed system, only communicate with their neighbouring nodes which are under the transmission range. To communicate with node which beyond the transmission range we make use Gossip Protocol as discussed in section II. We use deterministic gossip protocol to broadcast transactions, blocks and new locations over the network.

The following procedure demonstrate the working of the gossip protocol in our system.

Algorithm 2 - Inter node gossip communication algorithm

```
neighbours ≔ getNeighbours()
Function broadcast (<*,round,data>):
        for each neighbour in random(neighbours):
                send(neighbour, <*,round,data>)

upon <BLOCK,round,data> from neighbour ∈ neighbours do
        if valid(data,round)∧ updateChain(data) then
                broadcast <BLOCK,round,data>

upon <TRANSACTION,round,data> from neighbour ∈ neighbours do
        if valid(data,round)∧ updateTransactionPool(data) then
                broadcast <TRANSACTION,round,data>

upon <LOCATION,round,data> from neighbour ∈ neighbours do:
        if valid(data,round) ∧ updateRouter(data) then
                broadcast <BLOCK,round,data>
```

This algorithm is based on a probabilistic approach in which each node doesn't broadcast a received message further if they have the same message existing in their system already otherwise the nodes sends the message to its neighbours chosen on a random basis based on its closest neighbours.

The broadcast function takes a message as a parameter and sends it to the neighbours. The list of neighbours is constantly updated since the node is flying may break connections or make new ones, hence it might not send the data to the same node. Message handler for TRANSACTION, updates the transaction pool if the transaction and only a new transaction is broadcasted further. Similarly, the message handler for BLOCK works in a same manner where it updates the chain. The message handler for location constantly updates the location table of the flying node in the system, in this case the data contains the next location where the corresponding node will be. This approach is not only faster than peer to peer transmission but also concludes when all the nodes have received a message.

4.4. Consensus and leader election

In order to achieve consensus in the system, we use a simplified version of practical byzantine fault tolerance algorithm. The following algorithm demonstrates how consensus is achieved in our system.

Algorithm 3 - Consensus Algorithm

```
height ← 0
round ← 0
chain[]← nil
Function startRound(round):
        leader(height,round) ← electLeader(height,round)
        if leader(height,round) then
                proposedBlock ← getProposedBlock(height,round)
        else
                electLeader(height,round)

upon <PROPOSAL,height,round,proposedBlock> from leader(height,round) do
        if valid(proposedBlock) ∧ valid(round) then
                broadcast<PREVOTE,height,round,id(proposedBlock)>
```

```
upon <PROPOSAL,height,round,proposedBlock> from leader(height,round) AND 2f+1
<PREVOTE,height,round,id(proposedBlock)> only once do
        if valid(proposedBlock) ∧ valid(id(proposedBlock)) ∧ valid(round) then
                broadcast<PRECOMMIT,height,round,id(proposedBlock)>

upon <PROPOSAL,height,round,proposedBlock> from leader(height,round) AND 2f+1
<PRECOMMIT,height,round,id(proposedBlock)> only once do
        if valid(proposedBlock) ∧ valid(round) then
                chain[height] ← proposedBlock
                height  ← height + 1
                startRound(round + 1)
```

The first message that propagates during the consensus period is the PROPOSAL message. PROPOSAL message consists of a block with unconfirmed transactions ordered about a particular attribute. If the leader is not present or not responding, election is initiated again. Once the leader generates a proposal, it is broadcasted to the network using gossip protocol for the network's approval. Each node on receiving an uncommitted block, votes on the block if valid and broadcasts PREVOTE message and ignores otherwise. When the number of votes on the block is 2f+1 then the node saves the block and broadcasts a PRECOMMIT message to its neighbours. With 2f+1 PRECOMMIT messages to the for a block, nodes add the block to the chain.

Each round in the network requires a new leader to order the transactions and propose a block. Dynamic leader election ensures that the system is decentralized contrarily a static leader could a turn malicious. We have used a leader election algorithm similar to but a modified version of RAFT consensus algorithm which elects the new leader by starting a decentralized voting round. The following procedure demonstrates the working of the leader election process.

Algorithm 4 - Leader Election Algorithm

```
currentRound ←0
leader ←nil
startElectionRequests[] ←nil
grantedVotes[]← nil

Function beginElection(round):
        if valid(round) then
                currentRound ← round
                broadcast <START_ELECTION,round,request>

upon <START_ELECTION,round,request> do
        if valid(request> ∧ valid(round) then
                push(startElectionRequests,request)

upon 2f+1 <START_ELECTION,round,request> only once do
        broadcast <REQUEST_VOTE,startElectionRequests,round,request>

upon <REQUEST_VOTE,startElectionRequests,round,request> only once do
        if valid(round) ∧ valid(startElectionRequests) then
                broadcast <GRANT_VOTE,round,request>

upon <GRANT_VOTE,round,request> do
        if valid(request) ∧ valid(round) ∧ find(grantedVotes,data(request)) then
                push(grantedVotes,data(request))
                broadcast <GRANT_VOTE,round,request>

upon 2f+1 <GRANT_VOTE,round,request> only once do
        if valid(request) ∧ valid(round) then
                broadcast <APPEND_ENTRY,grantedVotes,round,request>

upon <APPEND_ENTRY,grantedVotes,round,request> only once do
        if valid(request) ∧ valid(grantedVotes) then
                leader ← id(request)
```

During a consensus round a new leader election process runs simultaneously. A node which wants to become the leader in the next round, participate in election undergoing in the previous round. The election process starts by broadcasting the START_ELECTION message to the network. With 2f+1 START_ELECTION representing 2f+1 node supporting the election, a node participates in the election process by broadcasting REQUEST_VOTE message along with a list of start election requests as a proof. When a node receives a valid REQUEST_VOTE it generates a GRANT_VOTE message and broadcasts it. The node that receives 2f+1 vote is elected as the next leader. The APPEND_ENTERY message tells the nodes to add the select the chosen as the leader for the next round. During the first round, after the addition of the genesis block, a static leader is chosen which bootstrap the network, but later a dynamic leader election procedure is followed.

5. Experimental Setup

This section provides the readers the experimental setup of this project. We implemented working model of the algorithms presented in the previous section in Node.js. Being light and event-driven, Node.js was a suitable choice to develop a prototype of this project. We visualized the flying nodes in a 2D environment and plotted its location using Vis.js, which is a graph visualization library for JavaScript. The visualization showed the successful generation of nodes, detection of neighbours, broadcast of transactions between nodes, generation of block and leader election. The experimental setup includes the parameters that are assigned to each node before the system is initiated. These parameters are namely, transmission range, speed, starting location, message size, block size, transaction threshold, routing protocol. For testing the above-proposed architecture while considering the low processing power of FANETs, we have considered the following parameters:

Table 1. Experimental Setup.

Parameter name	Assumed value
Block size	60 Transactions
Transactions Threshold	1000 Transactions
Starting location	Random
Transmission Range	1 Km
Flying Speed	10 Kmph
Message Size	100 KB
Transmission Protocol	Gossip

6. Result Analysis

This section describes three performance metrics [30] that have been evaluated for this paper. Following the definition of the metric, in context of blockchain and FANETS, we provide the results for each metric of our implemented architecture.

6.1. Throughput

Throughput, for this architecture, can be defined as the rate of committing of valid transactions. This rate has been expressed in transactions per second for given network size. Throughput has been measured by successively increasing the total nodes in the network. Following formula has been used to calculate throughput:

$$throughput = \frac{total\ commited\ transactions}{total\ time\ in\ seconds}$$

Table 2. Results.

Network Size	Avg. Transaction Timestamp	Block Timestamp	Throughput (transactions/sec)
5	1551805529299.43	1551805529517.00	275.7775601
10	1551805761821.48	1551805762027.00	291.9472176
15	1551808096519.15	1551808096736.00	276.6888309
20	1551805332010.2	1551805332214.00	294.4062101

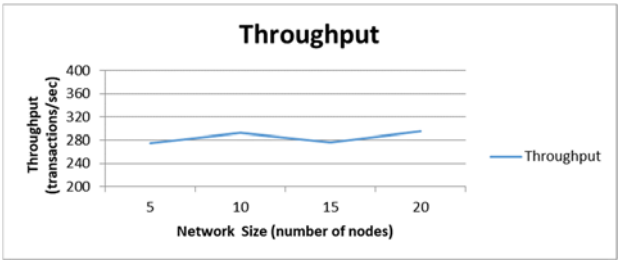

Fig. 4. Throughput

It can be inferred from the throughput graph that with the increase in the total number of nodes in the network the throughput varies by a small amount and remains approximately equal to 300 transactions per second.

6.2. Latency

Latency, for this architecture, can be defined as the time taken by the network to validate a transaction. This time includes the time from the point of submission to the point it is committed to the blockchain This rate has been expressed in transactions per second for given network size. Latency has been measured by successively increasing the total nodes in the network. Following formula has been used to calculate throughput:

$$latency = confirmation\ time - submission\ time$$

Table 3. Latency

Network Size	Avg. Transaction Timestamp	Block Timestamp	Latency (seconds)
5	1551805529299.43	1551805529517.00	0.21756665
10	1551805761821.48	1551805762027.00	0.205516602
15	1551808096519.15	1551808096736.00	0.216850098
20	1551805332010.2	1551805332214.00	0.203800049

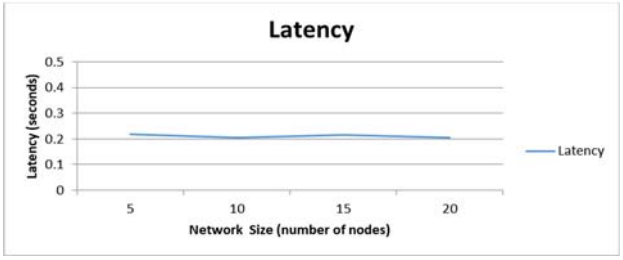

Fig. 5. Latency

It can be inferred from the latency graph that with the increase in the total number of nodes in the network, the latency remains nearly constant.

6.3. Message Overload

Message Overload, for this architecture, can be defined as the average of the total message broadcasted by the network until all the nodes in the network have received the particular transaction. This metric has been expressed in an average number of messages broadcasted for given network size. Message overload has been measured by successively increasing the total transactions submitted to the network.

Table 4. Message Overload

Transaction Count	Network Size	Total Messages	Message Overload
10	10	134	13.4
20	10	330	16.5
30	10	478	15.9
40	10	620	15.5
50	10	786	15.7
60	10	980	16.3

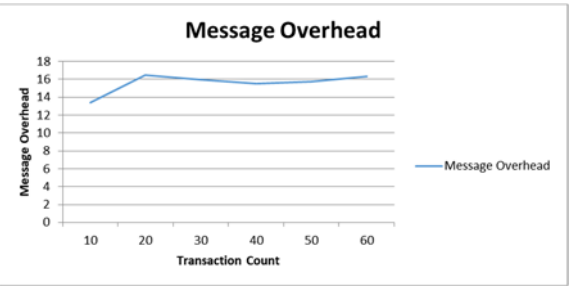

Fig. 6. Message Overload

It can be inferred from the message overhead graph that with the increase in the total number transactions submitted to the network the message overload, increases in the beginning but remains approximately constant afterwards.

7. Applications

The architecture proposed in this paper can applied in many areas. This section presents a few use cases that can adopt the proposed architecture as a cost-effective solution to its existing systems. Following are the possible use cases of a FANET running with a blockchain:

- In remote locations, providing good quality services is a challenge. This architecture can assist various businesses to increase the coverage area of their services.
- In the healthcare sector, this decentralized architecture will reduce the cost of providing such medical services by allowing businesses to share the cost of transportation and management.
- This architecture also allows the creation of a market of sensor data that would not be controlled by the large corporations but by the owner of the nodes.
- Communication services can be extended using this architecture by implementing specific routing algorithms and reaching out to remote locations.
- With the increase in the usage of drones and UAVs, air traffic is bound to increase. An architecture like this can support air traffic management.
- A decentralized architecture of UAV nodes can also be used to extend the functionality of existing FANETs and improve interoperability between them.

8. Conclusion

Blockchain and FANETs both are growing research areas and soon we will witness mass adoption of both these technologies in multiple industries. However, the security of FANETs and scalability of blockchain are major hindrances in its adoption. Our proposed blockchain-based architecture secures the exchange of messages and data exchanged between nodes using public-key cryptography, decentralizes the network by using blockchain and reduces major security risks that a single point failure could face. The experimental results have been shown that the proposed method works with constant through and latency with an increase in network size and approximately constant message overhead with increase transaction for given network size. While the given architecture is based on the context of flying nodes and their use case in extending smart cities, it is also applicable to other classes of ad hoc network nodes.

9. Future Scope

In future work, we will continue our research on blockchain and its application in flying ad hoc networks. We intend, to research more about the possible attacks and security threats that blockchain can prevent in FANETs. Furthermore, more research consensus algorithms used by IOTA, ripple, and Hashgraph, could be explored to extend this architecture. There still remains several open questions and challenges such as energy-efficient encryption, identity management, content distribution, and transmission, which require further research. Moreover, the paper is primarily based on a permissioned blockchain solution, algorithms such as Proof of stake and delegated proof of stake can be adopted to create a truly public network in the near future.

10. References

[1] D. N. Jesse, "Internet of Things and Big Data – The Disruption of the Value Chain and the Rise of New Software Ecosystems," *IFAC-PapersOnLine,* pp. 275-282, 2016.

[2] B. K. S, "Study of Ad hoc Networks with Reference to MANET,VANET, FANET," *International Journals of Advanced Research in Computer Science and Software Engineering,* 2017.

[3] Guy Zyskind, Oz Nathan, Alex 'Sandy' Pentland, "Decentralizing Privacy: Using Blockchain to Protect Personal Data," *IEEE CS Security and Privacy Workshops,* pp. 180-184, 2015.

[4] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System,* 2008.

[5] L. G. Daniel Fernando Pigatto, "HAMSTER – Healthy, Mobility and Security-based Data Communication Architecture for Unmanned Aircraft Systems," *International Conference on Unmanned Aircraft Systems (ICUAS),* 2014.

[6] Michael Nofer, Peter Gomber , Oliver Hinz ,Dirk Schiereck, "Blockchain Article," *Springer Fachmedien Wiesbaden 2017,* 2017.

[7] D. A. S. RAMASASTRI, "Application of Blockchain to Banking and Finance in India," An IDRBT Publication, 2017.

[8] "pluralsight," 10 Janunary 2019. [Online]. Available: https://www.pluralsight.com/guides/blockchain-architecture.

[9] Neha Garg, Partibha Yadav, "Comparison of Asymmetric Algorithms in Cryptography," *International Journal of Computer Science and Mobile Computing,* p. 1190 – 1196, 2014.

[10] Sachchidanand Singh, Nirmala Singh, "Blockchain: Future of Financial and Cyber Security," in *International Conference on Contemporary Computing and Informatics*, 2016.

[11] V. Buterin, *Ethereum White Paper.*

[12] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," ACMTCS, 2002.

[13] Ethan Buchman, Jae Kwon and Zarko Milosevic, "The latest gossip on BFT consensus," Tendermint, 2018.

[14] Mike Burmester, Tri Van Le and Alec Yasinsac, "Adaptive gossip protocols: managing security and redundancy in dense ad hoc networks," Department of Computer Science, Florida State University Tallahassee, Florida 323204-4530.

[15] B. Haeupler, "Simple, Fast and Deterministic Gossip and Rumor Spreading," *Microsoft Research,* 2014.

[16] Ali Dorri, Salil S. Kanhere, and Raja Jurdak, "Blockchain in Internet of Things: Challenges and Solutions," 2016.

[17] K. Christidis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE,* pp. 2292-2303, 2016.

[18] Mohamed Amine Ferrag, Makhlouf Derdour, Mithun Mukherjee,Abdelouahid Derhab,Leandros Maglaras,Helge Janicke, "Blockchain Technologies for the Internet of Things: Research Issues and Challenges," *IEEE,* 2018.

[19] Mandrita Banerjee, Junghee Lee, Kim-Kwang Raymond Choo, "A blockchain future to Internet of Things security: A position paper," *Digital Communications and Networks,* 2017.

[20] M. Arrivets, "MONET: Mobile Ad Hoc Blockchains," www.mosaicnetworks.io, 2018.

[21] Arne Bochem, Benjamin Leiding, and Dieter Hogrefe, "Unchained Identities: Putting a Price on Sybil Nodes in Mobile Ad hoc Networks," University of Goettingen, Institute of Computer Science, Goettingen,Germany, 2018.

[22] Mee Lan Han, Byung Il Kwak, Huy Kang Kim, "Anomaly Intrusion Detection Method for Vehicular Networks Based on Survival Analysis," *Vehicular Communications,* pp. 52-63, 2018.

[23] Sparsh Sharma and Ajay Kaul, "A Survey on Intrusion Detection Systems and Honeypot based Proactive Security Mechanisms in VANETs and VANET Cloud," *Vehicular Communications,* pp. 138-164, 2018.

[24] Sabri M. Hanshi, Tat-Chee Wan, Mohammad M. Kadhuma, Ali Abdulqader Bin-Salem, "Review of Geographic Forwarding Strategies for Inter-vehicular Communications from Mobility and Environment Perspectives," *Vehicular Communications,* pp. 64-79, 2018.

[25] Souaad Boussoufa-Lahlah, Fouzi Semchedine, Louiza Bouallouche-Medjkoune, "Geographic routing protocols for Vehicular Ad hoc NETworks (VANETs): A survey," *Vehicular Communications,* pp. 20-31, 2018.

[26] Xu Yanga, Xun Yi, Ibrahim Khalil, Yali Zeng, Xinyi Huang, Surya Nepal, Xuechao Yang, Hui Cui, "A lightweight authentication scheme for vehicular ad hoc networks based on MSR," *Vehicular Communications,* pp. 16-27, 2019.

[27] Benjamin Leiding,Parisa Memarmoshrefi,Dieter Hogrefe, "Self-managed and Blockchain-based Vehicular Ad-hoc Networks," UBICOMP/ISWC ’16 ADJUNCT, HEIDELBERG, 2016.

[28] Madhusudan Singh,Shiho Kim, "Blockchain Based Intelligent Vehicle Data sharing Framework," 2017.

[29] Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park, "Block-VN: A Distributed Blockchain Based Vehicular Network Architecture in Smart City," *JIPS,* pp. 184-195, 2017.

[30] Hyperledeger, *Hyperledger Blockchain Performance Metrics.*