Fusion of Named Data Networking and Blockchain for Resilient Internet-of-Battlefield-Things

Ronald Doku, Danda B. Rawat, Moses Garuba
Data Science and Cybersecurity Center (DSC2)
Department of Electrical Engineering and Computer Science
Howard University, Washington DC, 20059, USA
E-mail: danda.rawat@howard.edu

Laurent Njilla
Cyber Assurance Branch
US Air Force Research Laboratory
Rome, NY 13441, USA
E-mail: laurent.njilla@us.af.mil

Abstract*—Named Data Network's (NDN) data-centric approach makes it a suitable solution in a networking scenario where there are connectivity issues as a result of the dynamism of the network. Coupling of this ability with the blockchain's well-documented immutable trustworthy-distributed ledger feature, the union of blockchain and NDN in an Internet-of-Battlefield-Things (IoBT) setting could prove to be the ideal alliance that would guarantee data exchanged in an IoBT environment is trusted and less susceptible to cyber-attacks and packet losses. Various blockchain technologies, however, require that each node has a ledger that stores information or transactions in a chain of blocks. This poses an issue as nodes in an IoBT setting have varying computing and storage resources. Moreover, most of the nodes in the IoT/IoBT network are plagued with limited resources. As such, there needs to be an approach that ensures that the limited resources of these nodes are efficiently utilized. In this paper, we investigate an approach that merges blockchain and NDN to efficiently utilize the resources of these resource-constrained nodes by only storing relevant information on each node's ledger. Furthermore, we propose a sharding technique called an Interest Group and introduce a novel consensus mechanism called Proof of Common Interest. Performance of the proposed approach is evaluated using numerical results.

***Index Terms*—Internet-of-Battlefield-Things, IoBT, Blockchain for IoBT, NDN for IoBT.**

I. Introduction

The emergence of Internet-of-Battlefield-Things (IoBT) draws its inspiration from the work that is being investigated over in the Internet of Things (IoT) sector. The origins of IoT can be traced back to government-funded projects in the late 1970s. These projects were primarily aimed at developing sensors that could thwart enemy threats on the battlefield. Most of this research work became the bedrock on which some key concepts from the current IoT field were obtained. The direction the research work on IoBT is heading resembles something out of a Hollywood science fiction movie. For example, there is ongoing research work that could allow trees, rocks and other objects to act as data gathering devices that collect data on encroaching adversaries. This is however still far away from reality, but a more realistic way of using IoBT would be to provide soldiers with smart wearables that would allow them to collect both personnel and operational data. IoBT would enable combatants to know what they need to look out for and what they need to look for while accomplishing the objectives of an assigned mission. However, IoBT networks have high network disruptions and as such, requiring devices to coordinate seamlessly in such a dynamic network with a relatively high packet loss rate is an issue. This paper studies the integration of blockchain with NDN by leveraging their best features (named data, immutable communications, peer-to-peer decentralized ledger, sharding, etc.) for a more resilient IoBT network.

This work is partly supported by the U.S. Air Force Research Lab (AFRL), U.S. National Science Foundation (NSF) under grants CNS 1650831 and HRD 1828811, and by the U.S. Department of Homeland Security (DHS) under grant award number, 2017-ST-062-000003. However, any opinion, finding, and conclusions or recommendations expressed in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the funding agencies.

II. Background and Integration of NDN and Blockchain for IoBT

NDN provides several advantages over IP based protocols in networks where nodes are continually on the move, such as an IoBT setting [1]. This can be noticed in NDN's architecture as its design makes it a suitable solution for IoBT. A primary reason why this is so is in the way NDN handles data routing issues effectively in dynamic and disruptive settings. Data is propagated on the network through stateful forwarding, which is achieved without the need for host addresses (source and destination) [2] because unlike other protocols, NDN names the data found in the network [3]. The work in [4] studied NDN's ability to thrive in a dynamic environment. They point out how NDN's data-centric approach to storing data makes it better suited for conditions where nodes are always mobile. NDN's advantage in the IoT has been highlighted by [5], where NDN's ability to encrypt the data itself instead of at the end-hosts offers an advantage [6]. NDN attempts to propagate relevant data to nodes in the network. We take this approach a step further by letting nodes compute the relevance of the data being propagated in the network. A node only adds data to its ledger if only that specific data is more relevant than any of the data it currently has stored on its ledger. This ensures resource-constrained devices only store important data. NDN's approach is ideal for IoBT environments because it reduces the complexity of dealing with TCP/IP connections. Intermittent connections are the order of the day as IP addresses tend to change over time because TCP/IP requires both ends to be connected at the same time. IP based networks have a dependency on Infrastructure services such as DHCP and DNS services, which presents the arduous task of ensuring sustained connectivity in dynamic environments. As such, TCP connections are beset with high packet losses. The high packet loss rate is an issue our approach attempts to remedy. Furthermore, NDN's design does not require the necessity to secure end to end channels. The data is the only thing that is secured. NDN traffic is normally pull-based. There is no traffic unless a consumer requests explicitly for specific data. The consumer has to know (or assume) that there are certain data available somewhere to send an interest packet. In our approach, the consumer would belong to a group that shares somewhat similar data, and as such, sends interest packets to the members in that group. We call such a group an Interest Group (IG). We delve deeper into the formation of IGs later on in the paper. In the area of security, NDN secures data directly at the packet level [7]. Furthermore, the research done in Name Based Access Control (NAC) [6] ensures trust by enforcing access control policies based on data. It supports data confidentiality and access control, which has become a growing requirement in military communication. Burke et al. [8] compiles relevant work done in the domain of applying NDN in battlefield environments. They highlight the benefits NDN has over TCP/IP in such dynamic and mobile environments.

Bitcoin [9] was the first platform that opened the eyes of the world to the potential of the blockchain when it was used to solve the infamous double-spending problem in digital currency back in 2009. What Bitcoin brought to the table was the formation of a distributed peer to peer network where each node in the network had a copy of every transaction that had been performed on the network, which was called the blockchain. It also demanded that each node would have to solve a cryptographic puzzle that verifies transactions performed on the network. The solving of this puzzle meant there was some computational work that had to be completed. This was the underlying concept that made Bitcoin so powerful. The solving of this cryptographic puzzle is what is now known as the proof of work (PoW) [9]. The PoW enabled the extension, immutability, trust, security, and maintenance of Bitcoin's public ledger [10]. Data (or transactions) can only be appended to the blockchain after a node or collection of nodes has solved a puzzle, a process called mining. Integrity and validity are assured because of this mining process [11]. There are several blockchain applications and use cases [12], and the rules governing the transactions differ from each blockchain application. The rules embedded in its core of blockchain are used to determine the validity of transactions that are propagated on the blockchain network. Transactions are first created by individual nodes and then disseminated to other nodes, usually to the nearest neighbors of the node that performed the transaction. Each node gathers the propagated transactions for a given amount of time. This time frame is also blockchain technology-specific and thus varies. After the allotted time has passed, the block is then mined. The mined block is then appended to the blockchain after a consensus is reached. For example, in bitcoin, consensus is reached when the network approves on the mined block as legitimate and then adds the newly mined block to the blockchain. The longest blockchain on the network then becomes the de facto blockchain.

IoT's vision is to connect things. However, managing trust in IoT is still an ongoing problem [13]. In our approach, we introduce the blockchain technology in an attempt to remedy the trust issues in the IoT space. In doing so, we propose a network that propagates trusted secure data. Blockchain in the IoBT field is still in its infancy stage. Most blockchain technologies require nodes to store a full-size ledger. This is, however, not feasible in conditions where a node in the network has limited resources such as the IoBT. In our approach, we ensure a node in the network only stores relevant information on its ledger. It does not have to store the full distributed ledger, but only what it deems as essential/relevant. Our Proof of Common Interest (PoCI) approach ensures the relevance and validity of data. Each data packet in the network is secured through NDN's design. While NDN handles the confidentiality issue through securing the data at the packet level, the blockchain ensures data integrity and also addresses packet flooding issues by introducing a cluster-aware interest forwarding protocol we describe later on. To the best of our knowledge, the integration of NDN and blockchain for a resilient IoBT has not been investigated in a state-of-the-art work. Thus, our work attempts to leverage the best features of both technologies for improving the overall performance of the IoBT.

III. Proposed Approach

The purpose of this research is to introduce an NDN-blockchain based approach to data propagation in an IoBT network. In our network, a node adds data to its private ledger if the incoming data's similarity score is higher than any of the data it has stored in its ledger. Our approach is motivated by the fact that IoBT devices have varying resource capacities, and as such, some are plagued with limited resources. We assume such devices must strive to exhaust resources on data they only consider to be relevant. To achieve this, we introduce a novel consensus mechanism that requires nodes in the network to solve a computationally inexpensive calculation called the Proof of Common Interest (PoCI). The PoCI is simply a similarity score that determines whether a node adds a data packet to its ledger. The PoCI seeks to ascertain whether the data producer's interest criteria and private ledger contents align with what the data consumer deems to be relevant. The steps below describe how this process is achieved:

Step 1: IoBT node Z (with a preset similarity threshold) in need of data sends interest packets to members of its IG

Step 2: When a member node of the IG receives the interest packet, it checks its ledger to find the data with the highest similarity match

Step 3: If the data with the highest similarity match passes the preset similarity threshold, the data is sent to the IoBT node Z

Step 4: IoBT node Z then adds the data to its ledger

A key feature of our network is an Interest Group (IG). IGs are generated as a result of the introduction of clustering to our network. Devices are segmented into clusters based on the similarities they share with other nodes. We call these clusters IGs. The clustering of the network provides unique advantages we discuss later on in this paper. Fig. 1 depicts an example of how clustering might occur in the IoBT setting. An IoBT node can belong to multiple clusters in the network. This would be a common theme for large nodes (helicopters, drones, tanks, etc.) who have the required resource needed to provide a large space for ledger storage. Such large devices will have interests in many of the data being disseminated in the network because of their ability to store a large set of data. On the other hand, some devices have limited storage, and thus, must only store relevant data in order to utilize its resource capacity efficiently. IG clustering works by ensuring two nodes that share the same similarity are placed in the same cluster to ensure efficient data dissemination.

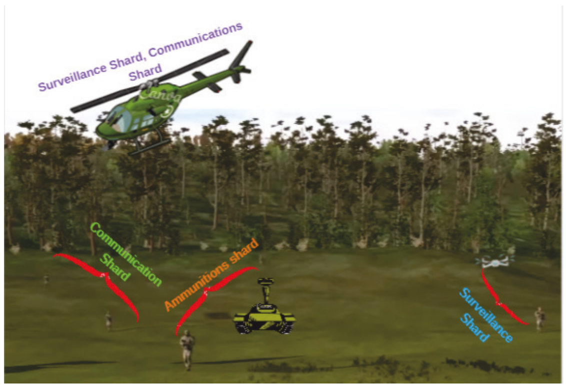

Fig. 1. A typical system model with clustering in an IoBT.

A. Space Allocation

Proof of Work (PoW) was a method proposed in [14] to resolve the problem of spam emails. It demanded that work was executed by any node sending an email. The PoW became a catalyst that fueled the development of the first-ever blockchain technology known as Bitcoin. However, PoW was beset with multiple difficulties, with the chief culprit being energy consumption issues. Various proof of X mechanisms, like the proof of stake, proof of burn, etc. are being investigated to be adopted in place of PoW [15]. Another proof of X mechanism which has recently gained traction is the Proof of Space (PoS), also known as the Proof of Capacity. It stems from the fact that there is a lot of unused storage in various devices, and as such, researchers have tried to take advantage of this opportunity and backed by the notion that the utilization of unused disk spaces as a consensus mechanism is better off than the energy-demanding PoW approach. This is because dealing with storage resources instead of computing resources is cheaper, as is evident in the large amount of it available in large computing devices [16].

However, PoS would not do comparatively well in an IoT/IoBT setting as nodes found in such networks do not have the luxury of having unused disk spaces. Thus, we propose a variant of the PoS in this work we call Limited Proof of Space (LPOS). In LPOS, we require that a node in our network dedicate at least 60% of its storage to the network at the initialization phase. This allocated space becomes the area in memory that will be allotted for the storage of the node's ledger. For example, consider an IoBT node *A* in the network. The network asks the node *A* for the disk space it is prepared to set aside at the initialization stage. If the node *A* has 60 MB of space and decides to dedicate 55 MB, it meets the requirement of the 60% of disk space utilization needed. This begins the initialization stage. The LPOS process proceeds by demanding the node generate random data packets of different sizes. The upper limit packet size is set to 500 bytes. Data packets that go above this range are split into multiple packets of that size. The ledger contains data packets that range from 50 bytes to 500 bytes. Each data packet is randomly assigned a similarity score at this step. These similarity scores are set to be of lower values in order to ensure that the actual data similarity scores surpasses it. Thus, after the initialization phase, each node's ledger would hold pseudo-randomly generated data (ranging from 50 - 500 bytes), each with varying similarity scores that will be replaced with actual data later on.

Data within the ledger are sorted in ascending order based on the similarity score. For actual data to be added, a data packet with a higher PoCI score must replace the data packet with the lowest score (top element) in the ledger. After replacement, the ledger is re-sorted again. This method ensures that only the data that a node finds relevant is appended to its ledger as it replaces data at the top of the ledger. The similarity score is calculated using the PoCI consensus approach. Fig. 2 depicts the process of replacing the data. In the figure, when node A performs the PoCI on the incoming data Packet Z, we observe that the PoCI score is 58%. It then replaces packet Z with the top element, packet A (which has the lowest similarity score). After replacement, the ledger gets re-sorted again. Each data packet is treated as a block in this scenario. The size of the block is the upper limit packet size (500 bytes). Unlike other blockchain networks where a block must contain multiple transactions, each transaction/data packet on this network is treated as a block. The PoCI process validates the data before it is added to the ledger which ensures data integrity.

B. Proof of Common Interest

In this section, we introduce the PoCI. The PoCI is a similarity function that produces a similarity score. Each node in the network is required to perform the PoCI on incoming data before it can add the data to its ledger. The PoCI attempts to find the common interests shared among nodes in a network. This is achieved by finding the similarity of the

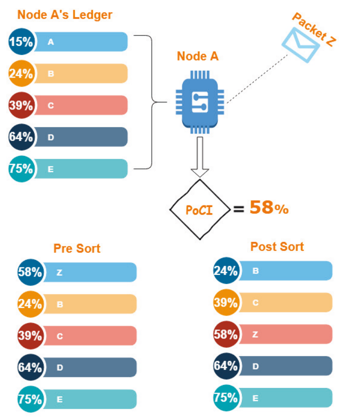

Fig. 2. Flowchart showing how ledgers are updated after PoCI is performed.

ledgers and the interest criteria of nodes in the network. Each node has an interest criterion, which is a list of all the topics it is interested in. To approach this problem, we find the intersection between two sets. To turn this into a set problem, we employ data mining methods such as shingling, Jaccard Similarity, MinHashing, and Locality Sensitive Hashing (LSH). The shingling of documents involves viewing a document as a set of short strings. In this manner, documents that share common sub-strings are perceived as similar. Shingling approaches this by transforming a document into multiple substrings of length k that is present within the document. Documents are represented as a set of k-shingles. The length k needs to be picked according to the size of the document. Picking k to be too small would result in a high presence of sub-strings of length k appearing in most of the documents being compared. For example, $k = 5$ is the common k length usually given to email-like documents with characters usually around 10,000. On the other hand, $k = 9$ is given to research articles with a lot more characters. Jaccard Similarity involves finding the similarity between sets (documents) by discovering the relative size of their intersection. When documents are presented as a set of shingles, we can use Jaccard Index to measure the similarity. The Jaccard Similarity between ledgers A and B is defined as:

$$JS(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

A larger shared number of shingles between ledgers represents a significant Jaccard Similarity. This applied to our approach means the ledgers of node A and node B are likely to be alike if their Jaccard Similarity is higher. Shingling, however, has a significant issue that needs to be solved. The document/information matrix that shingling produces tends to be a sparse matrix that brings along storage issues, especially in resource-constrained settings such as the IoBT. To address this, hashing has been introduced. Hashing works by converting a document of any size into a specific size. The size that is selected is usually small enough to fit into any memory. To solve the sparse matrix storage issue, the document/information matrix is transformed into a hash, and the Jaccard Indexes of the hashes is what is now calculated. To use the Jaccard index to find the similarity of two hashes, a special hash function named MinHashing is employed. A MinHash is a fixed-size numeric signature for each document. Consequently, utilizing MinHashing resolves the space complexity when having to compare two relatively huge ledgers for their similarity. We ideally only have to compare their signatures. MinHashing uses randomized algorithms to estimate the Jaccard Similarity between large documents. The steps below show the MinHashing process:

Step 1: Break down the ledger into a set of shingles.
Step 2: Calculate the hash value for every shingle.
Step 3: Store the minimum hash value found in step 2.
Step 4: Repeat steps 2 and 3 with different hash algorithms 199 more times to get a total of 200 min hash values (MinHash signature).

Using the steps above, we can compute the MinHash signature for a given ledger. This is accomplished by generating random hash functions of a certain quantity. For experimentation purposes, we chose 200 as the number of random hash functions to be generated. The initial hash function is applied to all of the shingle values in the ledger. We locate the minimum hash value produced and use it as the first component of the MinHash signature. We proceed to take the second hash function, and again find the minimum resulting hash value and use this as the second component. Because we have 200 random hash functions, we get a MinHash signature with 200 values (components). We use the same 200 hash functions for every ledger in the network and generate their signatures as well. 200 hashes are chosen arbitrarily and can be adjusted to lessen the work done by a node in order to reduce the computational work of the network, which is proportional to the energy consumption of the network. To compute the PoCI, we need to find the similarities between the ledgers and the interest criteria. The similarity score for the interest criteria can be computed by finding the Jaccard Similarity of the consumer node and the producer node. To find the similarity between two ledgers, we compare the ledgers by counting the number of signature components in which they match. That gives the similarity score for the comparison of any two ledgers. Nodes are required to calculate the MinHash signature anytime new information is added. The formula for calculating the MinHash of documents is expressed as:

$$h_\pi(C) = \min_\pi \pi(C) \tag{2}$$

where C represents a document.

To compute PoCI similarity function, we let $PoCI$ be the similarity score found by comparing the MinHashes of the ledgers and Jaccard Similarity of the Interest Criteria of node A and node B in the network. The following equation is used for the calculation:

$$PoCI = \alpha(JS[A, B]) + \beta(MH[A, B])) \tag{3}$$

where α and β are trust weights assigned to the result of the Jaccard Similarity between the two interest criteria of the nodes A and B, and the MinHash similarity between nodes A and B respectively. The network uses a reputation-based trust model to ensure an extra level of security. α represents the weight assigned to the node A and β represents the weight assigned to the node B. These weights are a reflection of the amount of trust a node has garnered in the network. To calculate these weights, we take into consideration these factors: the number of transactions a node has in the network, the number of PoCIs it has successfully passed, and the credibility of the nodes that ascertained the passing of the PoCI. To calculate α and β weights, we design a metric. We let U(k) be all the transactions a node *u* has, *P(k,i)* be the number of passed PoCIs during an i^{th} transaction, and *C(k,i)* be the credibility of the nodes that confirmed it. This can be calculated as a weighted average of the amount of relevant data a node has disseminated:

$$T(k) = \frac{\sum_{i=1}^{U(k)} P(k,i) * C(k,i)}{U(k)} \tag{4}$$

C. Interest Groups

The blockchain is facing scalability issues [17]. One of the techniques that has been employed to help to resolve the scalability issue is *sharding*. Sharding is simply segmenting a blockchain network into multiple groups called shards [18]. A shard has its ledger and can validate transactions [19]. By splitting the network in this manner, the network's efficiency is improved. These teams work together in parallel to maximize the performance of the network [20]. Various sharding techniques have been proposed. In our work, we present a new sharding technique we call an IG. The idea behind this approach is to place nodes with close similarity scores in the same shard. An advantage this offers is that it can curb unnecessary interest packet flooding as it consumes a lot of bandwidth and CPU processing power as nodes primarily send interest packets to members of its shard. We employ a self-learning technique to find packet delivery paths. Self-learning ensures producer nodes observe the paths to the consumer nodes that utilize the data and vice versa. In this paper, we introduce a self-learning cluster-aware forwarding protocol which ensures nodes multicast data packets to shards they know are more likely to have the data they require. This is useful as the usual routing announcements consume energy and time. The shards are created using the LSH algorithm [21]. LSH hashing operates by grouping similar documents in the same buckets, while documents that are not similar are likely to be placed in different buckets. We choose hierarchical clustering as our clustering algorithm of choice. LSH produces a set of candidate documents; we calculate the similarity of those candidates to determine the two most similar ledgers in the network at a time. This leads to immediate hierarchical clustering. At each level, we get the two most similar ledgers and merge them. We stop when we get to the desired number of clusters. We use the banding technique of LSH to determine efficiently the most probable nodes that can be grouped. LSH can be formally written as:

- U = Universe of objects
- $S : U \times U \rightarrow$ [0,1] = Similarity function

An LSH for a similarity S is a probability distribution over a set H of hash functions such that:

$$Pr_{h \in H}[h(A) = h(B)] = PoCI(A, B) \tag{5}$$

for each $A, B \in U$. We are trying to find if the probability of collision between A and B given a random hash function is equal to the similarity function PoCI (A,B). The *Algorithm 1* employs the agglomerative clustering method. During this process, each device *d* is assigned to its own cluster. The next step is to calculate the similarity between the clusters that has been assigned to each node. After this is done, the two clusters that are most similar are joined. We repeat this until we reach the desired number of clusters we want in the network. The related algorithm is shown below.

Algorithm 1 Hierarchical Cluster Shard Formation Algorithm

```
1: Given a set of nodes with ledgers $\{d_1, ..., d_n\}$
2: A similarity function sim$(c_1, c_2)$
3: for $i = 1$ to $n$ do
4:     $c_i$= $(d_i)$
5: C = $\{C_1, ..., C_n\}$
6: size = n + 1
7: while C.size $\geq$ numberOfClusters do
8:     $(C_{min1}, C_{min2})$ =min sim$(c_1, c_2)$ for all $c_i, c_j$ in C
9:     remove $c_{min1}$ and $c_{min2}$ from C
10:    add $c_{min1}, c_{min2}$ to C
11:    size = size + 1
```

IV. Performance Evaluation

The network has two data packets. A producer data packet and an interest data packet. The producer data packet has a unique naming convention. A producer data packet from a node with the name *GreyArrow* follows this naming convention. For example, this node will have a producer data packet with this naming convention: $SquadronFall/Surveillance/GreyArrow/$124744774/. In this naming convention, *SquadronFall* represents the name of the network. *Surveillance* is the name of the cluster the node belongs. *GreyArrow* is the device name and *124744774* is the MinHash signature (length 200 in our network) for *GreyArrow*'s ledger. *GreyArrow*'s interest packet also follows a similar naming convention: $SquadronFall/Surveillance/GreyArrow/$. The obvious difference is that the MinHash signature is missing from the name for the interest packet but can however be found in its contents. The data packet also contains the similarity threshold which is the similarity score of the top element of the node's ledger. The data packet also contains the interest criteria of the node. The PoCI can be performed using the Interest Criteria and the MinHash signature found in the Interest Packet.

To evaluate our proposed approach, we modeled an IoBT network and generated a set of 30 random nodes (drones, tanks, watches, etc) with varying memory capacity using

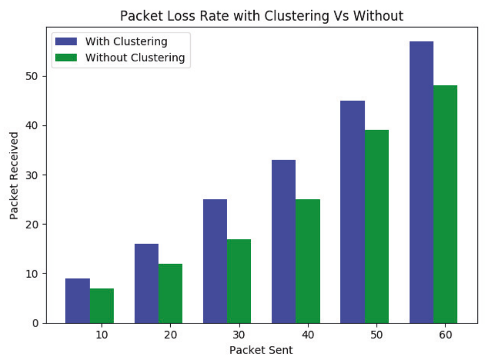

Fig. 3. Plots showing packet loss rate with and without clustering.

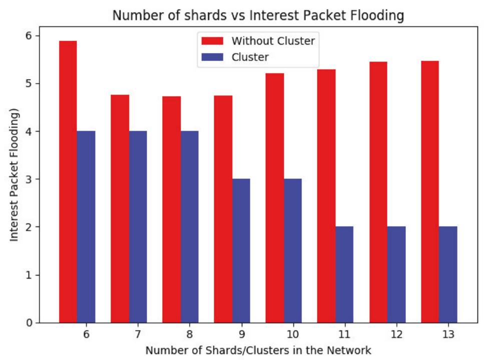

Fig. 4. Plots showing Interest packet flooding with and without clustering

PyNDN. We compared our approach against a normal NDN network. These experiments tested the packet retention features of both approaches. Since we focused on an IoBT network, we deemed packet retention as an important feature of such a network. From the plots, we can see from Fig. 3 the formation of clusters showed that there was a surge in data packet retention in the network compared to when there was no cluster formation (regular network). This showed that our approach (consequence of the introduction of IGs) helped reduce the packet loss rate. This is because the creation of self-learning clusters that accommodate nodes with similar interests ensures that nodes can effectively and efficiently transfer data among themselves which in turn leads to fewer packet losses. Fig. 4 shows that there is less packet flooding in the network as most of the data packets produced are multicast to clusters that share the same similar interests with the data producer's cluster.

V. Conclusion

In this paper, we devised an approach that merges the strengths of the blockchain and NDN in an IoBT network to get nodes to store the most relevant data that is being disseminated in the network. In our approach, we come up with a novel network sharding technique called the Interest Groups. With this, nodes with a closer similarity in terms of the data they store are assigned to the same Interest Group. Segmenting the network this way ensures that the network is not flooded with unnecessary Interest Packets. Nodes direct their Interest packets to their Interest Groups or other Interest Groups that are similar. Our paper also introduces a consensus mechanism we call the Proof of Common Interest. The PoCI is computed by finding the similarity between the interest of a data packet and the nodes in the IoBT network. The data packet is added to the ledger if the similarity meets a specified threshold. From the results of our evaluation, we can see that the segmentation of the network ensures effective packet transmissions and less packet flooding.

References

[1] B. Etefia, M. Gerla, and L. Zhang, "Supporting military communications with Named Data Networking: An emulation analysis," in *MILCOM 2012-2012 IEEE Military Communications Conference*, pp. 1–6, 2012.
[2] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.
[3] L. Zhang *et al.*, "Named Data Networking," *ACM SIGCOMM Computer Comm. Review*, vol. 44, no. 3, pp. 66–73, 2014.
[4] A. Afanasyev *et al.*, "A brief introduction to Named Data Networking," in *2018 IEEE Military Comm Conference (MILCOM)*, pp. 1–6, 2018.
[5] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, "Named data networking of things," in *2016 IEEE first Intl conference on internet-of-things design and implementation (IoTDI)*, pp. 117–128, 2016.
[6] Y. Yu, A. Afanasyev, and L. Zhang, "Name-based access control," *Named Data Networking Project, Tech Report NDN-0034*, 2015.
[7] Z. Zhang, H. Zhang, E. Newberry, S. Mastorakis, Y. Li, A. Afanasyev, and L. Zhang, "Security support in named data networking," tech. rep., Tech Report. Available online: https://named-data. net/wp-content, 2018.
[8] J. Burke, A. Afanasyev, T. Refaei, and L. Zhang, "NDN Impact on Tactical Application Development," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pp. 640–646, 2018.
[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009.
[10] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When Mobile Blockchain Meets Edge Computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018.
[11] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Pro. of the ACM Symposium on Principles of Dist. Computing*, pp. 315–324, 2017.
[12] D. B. Rawat, V. Chaudhary, and R. Doku, "Blockchain: Emerging Applications and Use Cases," *arXiv.org, arXiv:1904.12247*, 2019. https://arxiv.org/abs/1904.12247.
[13] L. Belli, S. Cirani, L. Davoli, A. Gorrieri, M. Mancin, M. Picone, and G. Ferrari, "Design and deployment of an IoT application-oriented testbed," *Computer*, vol. 48, no. 9, pp. 32–40, 2015.
[14] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual Intl Cryptology Conf*, pp. 139–147, Springer, 1992.
[15] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.
[16] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of Space," in *Annual Cryptology Conference*, pp. 585–605, Springer, 2015.
[17] Y. Sompolinsky and A. Zohar, "Accelerating bitcoin's transaction processing," *Fast Money Grows on Trees, Not Chains*, 2013.
[18] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, p. 8, 2013.
[19] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 17–30, ACM, 2016.
[20] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: A Fast Blockchain Protocol via Full Sharding,"
[21] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 253–262, ACM, 2004.