Article

Smart-Contract-Based Automation for OF-RAN Processes: A Federated Learning Use-Case

Jofina Jijin , Boon-Chong Seet * and Peter Han Joo Chong

Department of Electrical and Electronic Engineering, Auckland University of Technology, Auckland 1010, New Zealand
* Correspondence: boon-chong.seet@aut.ac.nz; Tel.: +64-9-921-9999 (ext. 5345)

Abstract: The opportunistic fog radio access network (OF-RAN) expands its offloading computation capacity on-demand by establishing virtual fog access points (v-FAPs), comprising user devices with idle resources recruited opportunistically to execute the offloaded tasks in a distributed manner. OF-RAN is attractive for providing computation offloading services to resource-limited Internet-of-Things (IoT) devices from vertical industrial applications such as smart transportation, tourism, mobile healthcare, and public safety. However, the current OF-RAN design is lacking a trusted and distributed mechanism for automating its processes such as v-FAP formation and service execution. Motivated by the recent emergence of blockchain, with smart contracts as an enabler of trusted and distributed systems, we propose an automated mechanism for OF-RAN processes using smart contracts. To demonstrate how our smart-contract-based automation for OF-RAN could apply in real life, a federated deep learning (DL) use-case where a resource-limited client offloads the resource-intensive training of its DL model to a v-FAP is implemented and evaluated. The results validate the DL and blockchain performances of the proposed smart-contract-enabled OF-RAN. The appropriate setting of process parameters to meet the often competing requirements is also demonstrated.

Keywords: smart contract; automation; opportunistic fog radio access network; industrial Internet-of-Things; federated deep learning; blockchain; computation offloading

check for updates

Citation: Jijin, J.; Seet, B.-C.; Chong, P.H.J. Smart-Contract-Based Automation for OF-RAN Processes: A Federated Learning Use-Case. *J. Sens. Actuator Netw.* **2022**, *11*, 53. https://doi.org/10.3390/jsan11030053

Academic Editor: Thomas Newe

Received: 7 August 2022
Accepted: 9 September 2022
Published: 13 September 2022

1. Introduction

Recent growth in big data and the use of artificial intelligence (AI) in the Internet-of-Things (IoT) led to an increasing need of resources for computation offloading and AI model training. However, advanced AI techniques such as deep learning (DL) are computationally resource-intensive if executed on IoT devices with low computation capacity. Hence, the need for them to offload their DL tasks to more resourceful devices increased significantly. Traditional offloading to the cloud via a cloud radio access network (C-RAN) has several issues, such as heavy workload at centralized baseband units (BBUs), limited backhaul capacity, and difficulty in serving delay-sensitive applications [1]. Consequently, researchers proposed fog radio access network (F-RAN), in which fog access points (FAPs) are deployed at the network edge to serve the IoT devices. These FAPs can be existing infrastructure entities further equipped with fog functionalities, or new entities deployed in an existing infrastructure [2]. However, existing F-RANs do not leverage the presence of available resourceful user devices to utilize their high, but idle, computation resources.

We argue that the DL tasks are more apt to be offloaded to an opportunistic F-RAN (OF-RAN), which we proposed in [3]. OF-RAN enhances the F-RAN by harnessing the concept of opportunistic networks (oppnets), a type of ad hoc network for utilizing available local resources in an opportunistic manner [4]. Each oppnet is established by a seed node that assigns one or more helper nodes to assist with a specific task. In OF-RAN, the role of the seed node and service node is equivalent to that of FAP in F-RAN, and helper node in oppnet. A seed node in the OF-RAN recruits locally available resourceful user devices, such as high-end smartphones and tablets, as service nodes, which collectively form a virtual FAP (v-FAP) to serve a resource-limited client, e.g., an IoT device.

In this paper, we consider an important problem that has yet to be addressed for OF-RAN to meet real-world deployment requirements, which is a trusted and distributed mechanism for automating its processes such as v-FAP formation and service execution. Automating these repetitive processes can improve operational efficiency, reduce the cost of service delivery, and help move towards a zero-touch network management model. The automation mechanism must be custom designed for the specific processes of OF-RAN, but such a mechanism has not been proposed for OF-RAN in the literature, to the best of our knowledge.

Motivated by the recent emergence of blockchain technology with smart contracts as an enabler of trusted and distributed systems, this paper proposes an automated mechanism using smart contracts for OF-RAN processes, built on our follow-up preliminary work on a blockchain-enabled OF-RAN in [5]. The system architecture of the proposed smart-contract-enabled OF-RAN is shown in Figure 1. At the access layer, seed nodes are infrastructure devices, such as Wi-Fi access points (APs) and pico- and femto-cell base stations (BSs) equipped with fog functionalities. Each seed node is a blockchain node, which hosts a smart contract, maintains a copy of the blockchain, and establishes a blockchain network with other seed nodes. At the terminal layer, each client is served by a v-FAP formed by multiple service nodes. The selection of service nodes in a v-FAP, placement of service tasks into service nodes, and processing of service tasks are all executed automatically, according to the smart contract.

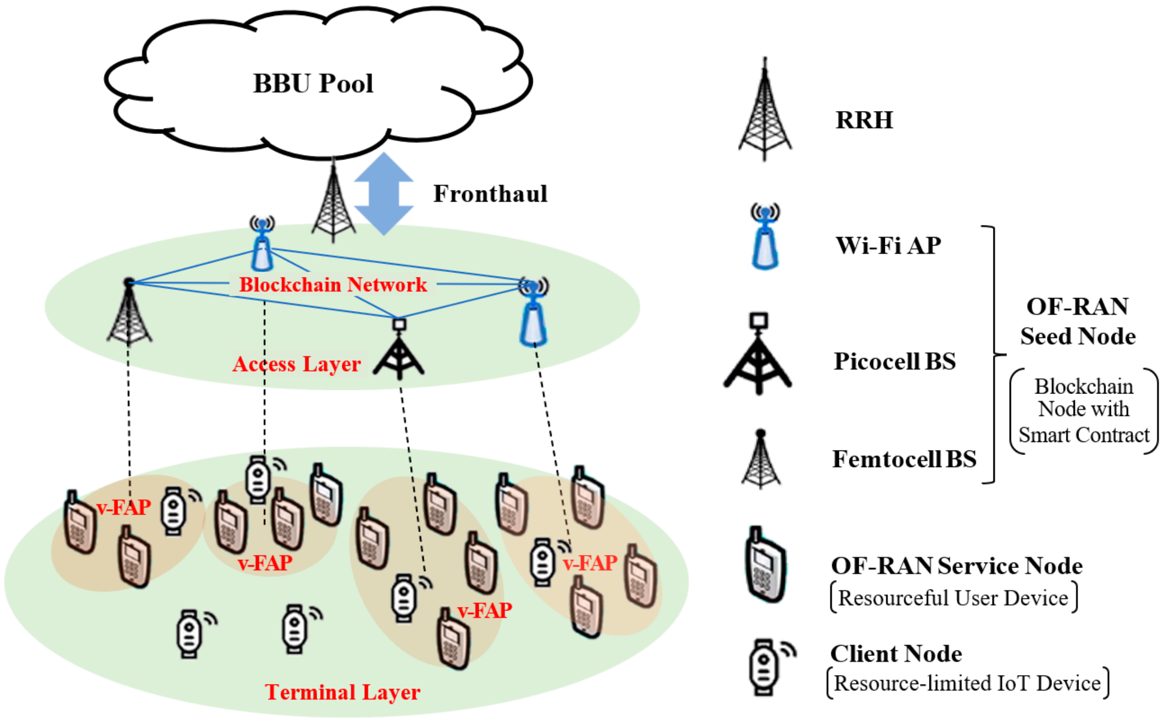

Figure 1. Smart-contract-enabled OF-RAN.

To demonstrate the role that our smart-contract-based automation for OF-RAN processes can play in real life applications, a federated DL use-case where a resource-limited client offloads the resource-intensive training of its DL model to a v-FAP is implemented on a physical testbed. The key contributions of this paper are:

- We propose a smart-contract-based mechanism for automating OF-RAN processes to provide trusted and distributed offloading services to resource-limited devices;

- We design four smart contracts for automating three OF-RAN processes and one application-specific (i.e., federated DL) process;
- We implement a v-FAP testbed to experimentally investigate our proposed system;
- We analyze the impact of various process parameters on the OF-RAN, blockchain, and federated DL performances.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 presents the system model. Section 4 details the process design for the proposed smart contract. The evaluation methodology and results are discussed in Section 5, and Section 6, respectively. Finally, Section 7 concludes the paper.

2. Related Works

Smart contracts and blockchain technologies are among the key enablers of Industry 4.0. This section reviews works on using blockchain with smart contracts for distributed systems to secure and automate their processes.

In [6], the authors proposed a blockchain-based secure DL for IoT, which supports collaborative DL with device integrity and confidentiality. The rules and policies to regulate the learning and mining tasks are defined in the form of smart contracts residing in the blockchain. The learning task is performed locally in IoT devices, and the learned local models are aggregated at an edge server acting as a blockchain node that mines and coordinates blockchain transactions. The proposed system is shown to be efficient in terms of accuracy, time delay, and security. However, due to limited resources of IoT devices, it is not suitable when large or complex learning tasks are involved.

The authors in [7] proposed blockchain-assisted federated learning for edge nodes to cooperatively train and predict popular files to be cached for IoT devices. Each edge node trains its local model, and then compresses and sends the local gradients to a cloud server for aggregation and update of the global model. The updated global model parameters are then returned to the edge nodes for further training or selecting files to be cached. In order to record, secure, and verify transactions, a smart contract constituting the following is proposed: (i) identity contract: verifies identity of IoT and edge nodes; (ii) submission contract: provides interface for edge nodes to submit their gradients to the blockchain; (iii) verification contract: elects supervisory consortium to verify transactions; (iv) credit contract: reward/penalizes participants. The proposed system is shown to improve cache hit rate and reduce file upload time. Although a blockchain is used, not much has been explored about the impact of blockchain parameters, such as block size and block interval, on the caching efficiency or security.

In [8], a security architecture for IoT networks based on software-defined networking (SDN), blockchain, and fog/edge computing is proposed. Decentralization in blockchain is used to secure sharing of IoT data and resources. A SDN-enabled edge switch continuously monitors the data flow to the fog nodes where traffic traces are learned and analyzed to identify malicious traffic flows. DL algorithms are used to detect attacks at network edge. A central cloud server manages the attack detection model in the fog nodes, which can be a processing or proofing agent. The processing agent trains the local model using local data obtained from the edge switch, while the proofing agent aggregates local models obtained from proofing agents and verifies the resulting attack detection model. All three entities (manager, processing agent, and proofing agent) interact with each other through transactions effectuated using a smart contract residing in the blockchain. The authors show that their architecture performs well in mitigating attacks, but do not evaluate its performance impact on delay-sensitive applications.

The authors in [9] presented EdgeChain, a blockchain-based architecture to make mobile edge application placement decisions for mobile hosts of multiple service providers (SPs). It uses the logic of the placement algorithm as a smart contract with the consideration of resources from all mobile edge hosts participating in the system. However, the proposed algorithm only considers fairness in resource sharing among multiple SPs. Other factors such as energy consumption and end-to-end latency, which are important to energyconstrained mobile devices and delay-sensitive applications, have not been considered. To reduce the blockchain's energy and computation requirements without compromising its traceability and non-repudiation, a lightweight blockchain system known as LightChain is proposed in [10]. It features a consensus mechanism with low computing power consumption, a lightweight data structure for information broadcast, and a method to limit the growing storage cost of the ledger.

In [11], a secure federated learning technique called Deepchain is proposed, where blockchain cryptographic features are used to preserve privacy of local gradients and guarantees auditability of training process. Its smart contract comprises a trading contract and processing contract, which guide the secure training process. It is evaluated in terms of cipher size, throughput, accuracy, and training time. Similarly, in [12], blockchain is employed to secure federated learning, but with the additional use of digital twins of end devices at edge servers to mitigate the issue of unreliable transmission links. However, it is unclear how deviations in data between end devices and their digital twins can impact the resulting edge intelligence. Furthermore, using blockchain to secure federated learning process can incur high mining costs and long information exchange delays due to the consensus protocol of the blockchain network. In contrast, our proposal herein does not use blockchain to secure federated learning, but information about the resourceful user devices in the OF-RAN, in order to facilitate their selection as service nodes for a v-FAP to perform federated learning or other offloading services.

3. System Model

Figure 2a shows the system model of the proposed smart-contract-enabled OF-RAN, in which the seed node and service nodes that constitute a v-FAP manage and execute the computation tasks offloaded by a client, respectively. The following explains the function of each key entities in the model:

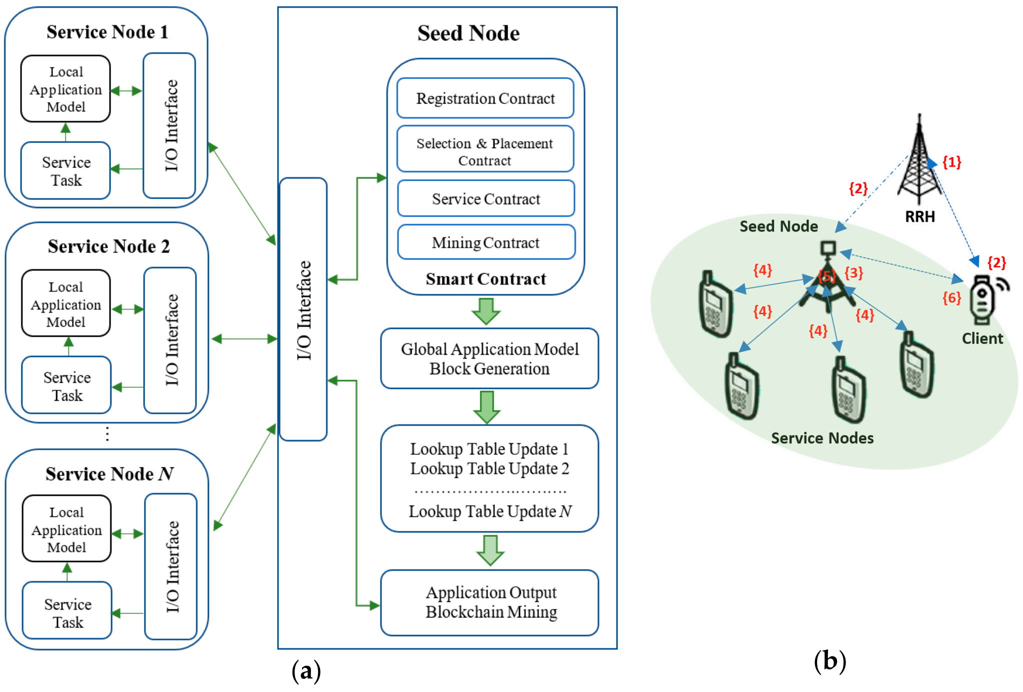

Figure 2. (**a**) System model; (**b**) Sequence of operations.

- *Smart Contract*: Defines the rules and logic for automating the OF-RAN processes through four sub-contracts: (i) registration; (ii) selection and placement; (iii) service; and (iv) mining. The registration contract registers interested resourceful user devices as potential service nodes. The selection and placement contract firstly selects a set of user devices based on the cost of using their resources as service nodes in a v-FAP, and then executes OF-RAN's task-to-node assignment (TNA) as defined in our follow-up work in [13]. The TNA is a process for the placement of the service tasks into the service nodes based on performance criteria such as node energy, process latency, and fairness in workload distribution. The service contract implements the service logic, which is application-specific. As a use-case of our proposed smart contract for OF-RAN, the federated learning application is chosen. The mining contract is responsible for new block generation from the transaction data generated upon executing the service contract to update the ledger;

- *I/O Interface*: For both seed node and service nodes to exchange information when serving a client;
- *Local Application Model*: A service node's application model that processes information from the seed node and generates a local outcome for the client;
- *Global Application Model*: A seed node's application model that collates the local outcome from each service node and generates a global outcome for the client;
- *Lookup Table*: Records the identity and performance of each service node, which can be looked up for future selection of service nodes when a new v-FAP is to be formed;
- *Application Output*: The global outcome generated by the global application model. In federated learning use-case, the application output is the aggregated weight, also referred to as global update for the client's DL model.

Figure 2b shows the sequence of operations of our system. For a resource-limited client to offload its task to OF-RAN, it first sends a service request {1} including the task requirements to its associated remote radio head (RRH), which in turn notifies the client {2} of an available nearby seed node to offload its task. The client then offloads its task {3} to this seed node in the form of data and initial model parameters. The seed node splits the task into sub-tasks, and, based on the TNA scheme, places the sub-tasks into each service node {4}. Upon processing, the service nodes send their local outcomes {5} to the seed node for collation. Finally, the seed node generates and returns a global outcome {6} to the client.

In our proposed smart-contract-enabled OF-RAN, every seed node is a blockchain node that monitors the transactions between nodes in a v-FAP. On completing the client's task, the seed node updates its lookup table, and then mines a new block from it as proof-of-work for propagation to blockchain under a permissioned consensus protocol [14]. Thus, computations performed in the v-FAP for an offloading application are unaffected by the blockchain computation performed by the seed node. This ensures that the delay-sensitive applications can be supported.

4. Process Design

This section details the design of our smart contract, which is constituted of four components for automating relevant OF-RAN processes: (i) registration contract; (ii) selection and placement contract; (iii) service contract; and (iv) mining contract, whose pseudo-code are shown in Algorithms 1–4, respectively.

4.1. Registration Contract

The purpose of the registration contract is for each seed node to maintain a registry of resourceful user devices in its proximity that could potentially become service nodes in a v-FAP to serve an offloading client. In this contract, the seed node broadcasts a request for expression-of-interest (EoI) in participating in a v-FAP. Each interested user device q_j replies with its identification ID_j and unit cost $u_{j,k}$ of using resource k for all K resources in q_j. The types of resources may include energy, computation, communication, and storage resources, as well as dwell time, i.e., amount of time user device remains available for the v-FAP or remains in coverage of the seed node, whichever is less. The seed node then registers the details for each replying device in its lookup table.

4.2. Selection and Placement Contract

The purpose of this contract is two-fold: (i) select N of registered user devices as service nodes in a v-FAP; (ii) execute the OF-RAN's TNA process in [13] for the placement of service tasks into service nodes. The selection is based on evaluating the total cost v_j of using each user device l_j in lookup table L to perform all M tasks from the client. Each task $m \in M$ has a demand R_k^m for resource $k \in K$. Knowing unit cost $u_{j,k}$ of using resource k in l_j, the cost $c_{j,k}$ of using resource k in l_j, for all M tasks can be calculated. Next, the total cost v_j of using all K resources in l_j, for all user devices in L, is calculated and ranked in ascending order of cost. Then, the top N service nodes, i.e., the N lowest-cost nodes, are selected.

Finally, the TNA process is executed to find the optimal placement of service tasks into service nodes based on the pareto-optimization of a multi-objective TNA problem [13]. The obtained solutions (placements) are non-dominant, i.e., not dominated by any member of the solution set for all the objectives. The seed node can select from the obtained placements, one that best meets the client's requirements, such as minimum completion time for delay-sensitive applications, maximum fairness for high-reliability applications by minimizing service node failures due to overloading, and minimum energy consumption for energy-conserving applications.

Algorithm 1: Registration Contrac

```
1.  Input: $Q = \{q_1, q_2, ..q_V\}$: set of $V$ user devices in proximity of seed node
2.  $\eta$: set of user devices that express interest as service nodes ($\eta \in Q$)
3.  $L$: lookup table of seed node
4.  $ID_j$: identification of user device $q_j$
5.  $u_{j,k}$: unit cost of using resource $k \in K$ in user device $q_j$
6.  $K$: set of resource types in a potential service node
7.  Output: registered user devices in $L$ as potential service nodes
8.  Process: broadcast request for EoI to all user devices in $Q$
9.  add to $\eta$ for each received EoI
10. if $|\eta| > 0$ // if not empty set
11.     for each $q_j$ in $\eta$ do
12.         register $ID_j$, $u_{j,k}$ in $L$ $\forall$ $k \in K$
13.     end for
14. end if
```

Algorithm 2: Selection & Placement Contract

```
1.  Input: $L = \{l_1, l_2, ..l_Y\}$: set of $Y$ registered user devices in lookup table
2.  $N$: number of service nodes in a v-FAP
3.  $M$: set of service tasks offloaded from the client to the v-FAP
4.  $K$: set of resource types in a potential service node
5.  $R_k^m$: demand for resource $k \in K$ by a service task $m \in M$
6.  $u_{j,k}$: unit cost of using resource $k \in K$ in user device $l_j$
7.  Output: TNA for v-FAP
8.  Process: for each user device $l_j$ in $L$ do
9.      for each resource $k$ in $K$ do
10.         $c_{j,k} = u_{j,k} \sum R_k^m \; \forall \; m \in M$ // cost of using resource $k$ in $l_j$ for all $M$ tasks
11.     end for
12.     $v_j = \sum c_{j,k} \; \forall \; k \in K$ // total cost of using all $K$ resources in $l_j$
13. end for
14. rank user devices in $L$ in ascending order of cost
15. $S \leftarrow$ top $N$ user devices in $L$ // select $N$ least-cost nodes as service nodes
16. execute $TNA(S, M)$
```

Algorithm 3: Service Contract for Federated DL Process

```
1.  Input: $S = \{s_1, s_2, ..s_N\}$: set of $N$ service nodes in a v-FAP
2.  $T$: total number of iterations in an epoch
3.  $P$: total number of epochs
4.  $\varepsilon$: termination threshold
5.  $w_c$: initial weights from client
6.  $L$: lookup table of seed node
7.  Output: final global weight $w^f$; updated $L$
8.  Process: initialize $w^f$, $w^{(p=0)}$, $w_j^{(t=0,p=1)} \leftarrow w_c$ for each $s_j$ in $S$
9.  set $p \leftarrow 1$
10. while $\left( \left| w^f \right| - \left| w^{(p-1)} \right| \leq \varepsilon \;\middle|\; p \leq P \right)$ do
11.   set $t \leftarrow 0$
12.   while $(t < T)$ do
13.      set $t \leftarrow t + 1$
14.      compute local update $w_j^{(t,p)}$ using (5) for each $s_j$ in $S$
15.   end while
16.   for each $s_j$ in $S$ do
17.      set $w_j^{(p)} \leftarrow w_j^{(T,p)}$
18.      envelop $Tx_j^{(p)} \leftarrow \left[ w_j^{(p)}, t_j^{(p)} \right]$ // create transaction
19.      upload $Tx_j^{(p)}$ to seed node
20.   end for
21.   $Tx^{(p)} = \left\{ Tx_1^{(p)}, Tx_2^{(p)}, ..Tx_N^{(p)} \right\}$ // seed node collates transactions
22.   verify $Tx^{(p)}$
23.   extract $w_j^{(p)}$ and $t_j^{(p)}$ from $Tx_j^{(p)}$ for each $Tx_j^{(p)}$ in $Tx^{(p)}$
24.   compute global update $w^{(p)}$ using (6)
25.   update $w^f \leftarrow \underset{w \in \{w^f, w^{(p)}\}}{\arg\min} F(w)$
26.   update $L$ with $t_j^{(p)}$ for each $s_j$ in $S$
27.   set $p \leftarrow p + 1$
28. end while
```

Algorithm 4: Mining Contract

```
1. Input: $S = \{s_1, s_2, ..s_N\}$: set of $N$ service nodes in a v-FAP
2. $L$: lookup table of seed node
3. $\beta$: block generation time
4. $\varphi$: hash pointer to previous block
5. Output: new block appended to blockchain
6. Process: retrieve $t_j^{(p)}$ from $L$ for each $s_j$ in $S$
7. $t^{(p)} = \left\{ t_1^{(p)}, t_2^{(p)}, ..t_N^{(p)} \right\}$
8. generate block $B^{(p)} \leftarrow \left\{ t^{(p)}, \beta, \varphi \right\}$
9. append $B^{(p)}$ to blockchain
```

4.3. Service Contract

The design of the service contract is application-specific. In this work, federated learning is used as an application example of the OF-RAN for computation offloading.

Federated learning is a new paradigm for machine learning, where DL models are executed locally in a distributed manner and results are sent to a server for aggregation [15]. The federated DL approach can be enabled by our smart-contract-enabled OF-RAN, where a resource-limited client can offload the training of its DL model by sending its training data and model parameters to a seed node in proximity. In turn, the seed node splits and sends the training data and model parameters to each service node in the v-FAP. The service nodes then train their respective local models, and send parameters of their trained models to the seed node. Finally, the local parameters are aggregated into a global model for returning to the client. Federated DL using service nodes in a v-FAP relies on collating their model's weight parameters obtained by learning from training data sets. A training data sample i is described as a two-dimensional array (x_i, y_i), wherein the DL model takes vector x_i as an input (e.g., image pixels) and gives a scalar output y_i (e.g., image label). For each sample i, the DL model with weight w computes a loss function $f_i(w)$, the result of which indicates the extent of model errors, and, thus, should be minimized in the learning process.

We consider a seed node with N resourceful user devices in its proximity that can be recruited as service nodes, and M is the set of training data from client. We denote the set of service nodes in a v-FAP as $S = \{s_1, s_2, \ldots s_N\}$. Each s_j, $j = \{1..N\}$ receives a subset of training data m_j from the seed node, and $M = \sum_{j=1}^{N} m_j$. A loss function for s_j over its training data m_j can be defined as:

$$F_j(w) \triangleq \frac{1}{|m_j|} \sum_{i \in m_j} f_i(w) \tag{1}$$

where $|m_j|$ returns the size of m_j. The global loss function for a v-FAP can be defined as:

$$F(w) \triangleq \frac{\sum_{j=1}^{N} F_j(w)}{|M|} \tag{2}$$

The goal of DL task is to find optimal weight w' parameters that minimize $F(w)$:

$$w' \triangleq \arg\min F(w) \tag{3}$$

We denote $w_j^{(t)}$ as the local model parameters of each service node s_j at iteration t of learning. Here, $t = 0,\ 1,\ 2, .., T$, where T is the maximum number of iterations. Each s_j trains its local DL model using the subset of training data m_j. At the beginning, all service nodes in S initialize their local model parameters. At each subsequent iteration $t > 0$, each s_j updates its $w_j^{(t)}$ by minimizing the loss function using the gradient descent update rule in (4), where $\lambda > 0$ is the learning rate and $\nabla F_j\left(w_j^{(t-1)}\right)$ is the average gradient on its training data at the previous local model parameters $w_j^{(t-1)}$:

$$w_j^{(t)} = w_j^{(t-1)} - \lambda\, \nabla F_j\left(w_j^{(t-1)}\right) \tag{4}$$

After T iterations, the updated local model parameters from each s_j are sent to the seed node where they is aggregated once every P epochs into a global model update. For each epoch, a total of T iterations of local update are performed at each s_j. The local update of s_j at epoch p and iteration t is given by:

$$w_j^{(t,\,p)} = w_j^{(t-1,p)} - \frac{\lambda}{m_j}\left(\left[\nabla F_j\left(w_j^{(t-1,p)}\right) - \nabla F_j\left(w_j^{(p-1)}\right)\right] + \nabla F\left[w^{(p-1)}\right]\right) \tag{5}$$

where $p = 1, 2, ..P$, $w^{(p)}$ is the global update at epoch p, and $\nabla F\left(w^{(p)}\right) = \frac{1}{|M|}\sum_{j=1}^{N} m_j \nabla F_j\left(w^{(p)}\right)$ is the global gradient value at epoch p after T iterations. Let $w_j^{(p)}$ be the local update of s_j at epoch p after T iterations. Then, w^p is updated as:

$$w^{(p)} = w^{(p-1)} + \frac{1}{|M|}\sum_{j=1}^{N} m_j\left(w_j^{(p)} - w^{(p-1)}\right) \tag{6}$$

The final output of this process is $w^{(f)}$, which gives the final model update that produces minimum global loss over an entire execution of local and global updates.

At the end of every epoch p, each s_j in S envelops its local update $w_j^{(p)}$ and task completion time $t_j^{(p)}$ to create a transaction $Tx_j^{(p)}$ to upload to the seed node. Upon receiving, the seed node verifies the transactions, extracts the local updates $\{w_1^{(p)}, w_2^{(p)}, ..w_N^{(p)}\}$ to compute the global update $w^{(p)}$, and updates its lookup table by recording the completion times $\left\{t_1^{(p)}, t_2^{(p)}, ..t_N^{(p)}\right\}$ of each service node. The completion times can be utilized by the seed node for a rating system that could affect their future selection for the v-FAP.

4.4. Mining Contract

The purpose of this contract is to generate and append a new block to the blockchain. Each block comprises a body and header. The body contains the collated service nodes' completion times $t^{(p)} = \{t_1^{(p)}, t_2^{(p)}, ..t_N^{(p)}\}$, while the header contains information about the seed node's block generation rate β (a.k.a. block interval time) and hash pointer φ to the previous block. On retrieving and collating the completion times from its lookup table, the seed node generates a new block $B^{(p)} = \left\{t^{(p)},\ \beta, \varphi\right\}$ for appending to the blockchain.

5. Evaluation Methodology

5.1. Emulation

This section describes our emulation of a smart-contract-operated v-FAP to assist with client training of a DL model based on federated learning for image-based object detection. The v-FAP is emulated using four Raspberry Pi 4 Model B single-board computers (4 GB RAM, 1.5 GHz CPU) as service nodes, and an Acer Aspire F15 laptop (8 GB RAM, 2.5 GHz CPU) as seed node (see Figure 3). The laptop is configured as a WiFi hotspot to communicate with the service nodes. The DL models in both seed and service nodes are implemented using Python 3.7 and TensorFlow 2.3.0. Python is also used for implementing the smart contract in the seed node.

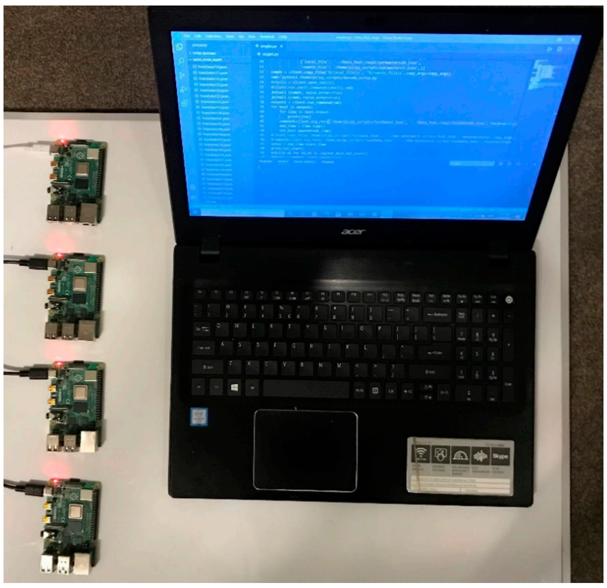

Figure 3. Emulated v-FAP for offloaded DL tasks.

For federated learning, we use the MNIST dataset [16], which contains 33,600 training instances and 8400 validation instances of 10 object classes. The training dataset (33,600 rows × 785 columns) is split into 1050 mini batches (each of 32 rows × 785 columns) for equal distribution among the service nodes. The DL model of each service node is a deep neural network (DNN) with seven layers: one input, one output, and five hidden layers. Each service node trains its own DNN model and sends the trained model parameters to the seed node. Two metrics used in this emulation evaluation are defined as follow:

- *Mean precision accuracy (MPA)*: The percentage of correctly predicted test instances using the global model from the total number of test data instances;
- *Latency*: The total time incurred for one epoch operation of the federated DL process. This includes both computation and communication time.

For the two metrics above, the number of service nodes in the v-FAP is considered the most important factor underlying their performance. In Section 6.1, we discuss the results of these metrics under the effect of varying number of service nodes.

5.2. Simulation

We use a realistic simulator to evaluate the performance of our blockchain network for OF-RAN. The Bitcoin simulator [14] built on the ns3 network simulator is used for simulating a realistic blockchain network with a set of consensus and network parameters such as network delay, block generation time, and block size. The following defines some parameters and metrics used in our simulation evaluation:

- *Block interval*: The time interval between blocks being added to the blockchain. Herein, the interval depends on the average time for service nodes to compute and send their transactions to a seed node for a new block to be generated;
- *Stale block*: Refers to a block not added to the blockchain due to concurrency or conflicts between miners. It triggers chain forks that slow the growth of main chain and, thus, is detrimental to the security of the blockchain;
- *Stale block rate*: The percentage of stale blocks among the total number of blocks mined;
- *Throughput*: The number of transactions in a block per unit of block interval time, in units of transactions per second (tps).

Table 1 shows the simulation settings used. The block size is set as recommended in [14]. The block interval is selected based on our findings on the time incurred by our emulated v-FAP with 1–4 service nodes. The transaction size used reflects the approximate size of information sent by service nodes to the seed node. The number of miners, i.e., seed nodes, is set to 16, to adequately represent the scale of the blockchain network that we envisioned for OF-RAN, while keeping the simulation time tractable.

Table 1. Simulation settings.

Parameter	Value	Unit
Block size δ	0.25–4	MB
Block interval τ	2–7	minutes
Transaction size	1	KB
Number of miners	16	

As mentioned above, the stale blocks can weaken the security of the blockchain, and parameters such as block size and block interval are known to affect the stale block rate. Additionally, the block size controls the system throughput. Thus, an appropriate block size and block interval are required for our blockchain to balance a trade-off between security and throughput. In Section 6.2, we discuss the results of stale block rate and throughput under the effects of varying block size and block interval.

6. Results and Discussion

6.1. Effect of Varying Service Nodes

The number of service nodes N in a v-FAP can impact training of the federated DL model. Figure 4 shows the results obtained from our emulated v-FAP in terms of latency and mean precision accuracy (MPA) under varying N. It can be observed that latency decreases as N increases. This is firstly because higher N splits the training data into smaller mini-batches, resulting in each service node incurring smaller computation delay. Furthermore, since the learning tasks in all service nodes are executed in parallel, increasing N decreases the maximum delay among the service nodes. It is also observed that the overall latency is significantly dominated by computation rather than communication delay.

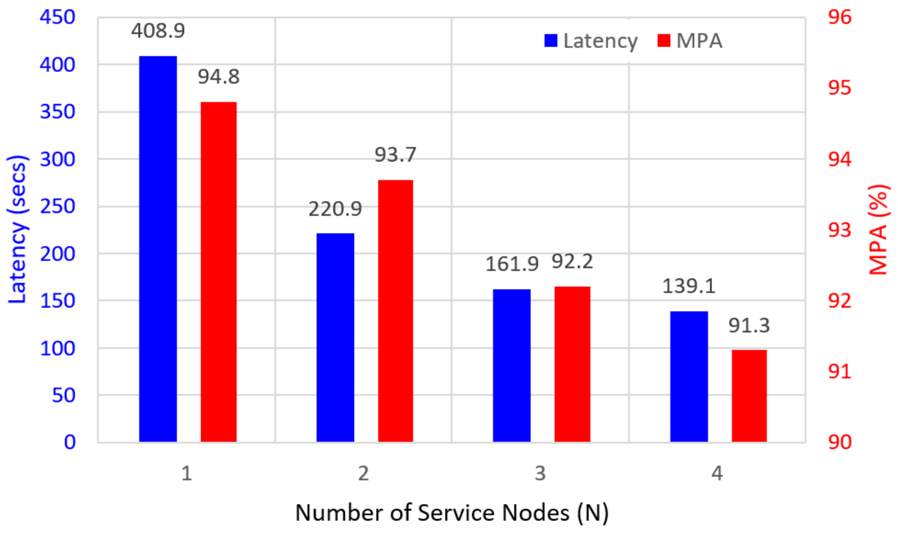

Figure 4. Effect of varying N on latency and mean precision accuracy.

The MPA is determined in the seed node using the global model obtained after the aggregation of local updates from each service node in the v-FAP. The results show that as N increases, the MPA expectedly decreases, but only marginally, which can be attributed to the reduced number of mini-batches per service node. Hence, it can be inferred that the MPA depends on the number of mini-batches used for training the local model of the service nodes.

There is an inherent trade-off between latency and the accuracy of training the global model. The seed node can, thus, select the number of service nodes in a v-FAP based on the client's requirements. For instance, when a learning task is delay-sensitive, the seed node can use more service nodes to reduce latency. However, if high accuracy is more critical than low latency, the seed node can use either fewer service nodes or more training epochs to improve accuracy.

6.2. *Effect of Varying Block Size and Block Interval*

The setting of block size δ and block interval τ can impact the security and throughput of our blockchain network for OF-RAN. Their settings are, in turn, factors to consider when determining the appropriate number of service nodes N in a v-FAP.

Figure 5 shows the stale block rate and throughput under varying δ. The results are obtained for a default τ = 4.5 min. It can be seen that as δ increases, the throughput increases. However, the stale block rate also increases, which can cause the blockchain to be more susceptible to attacks.

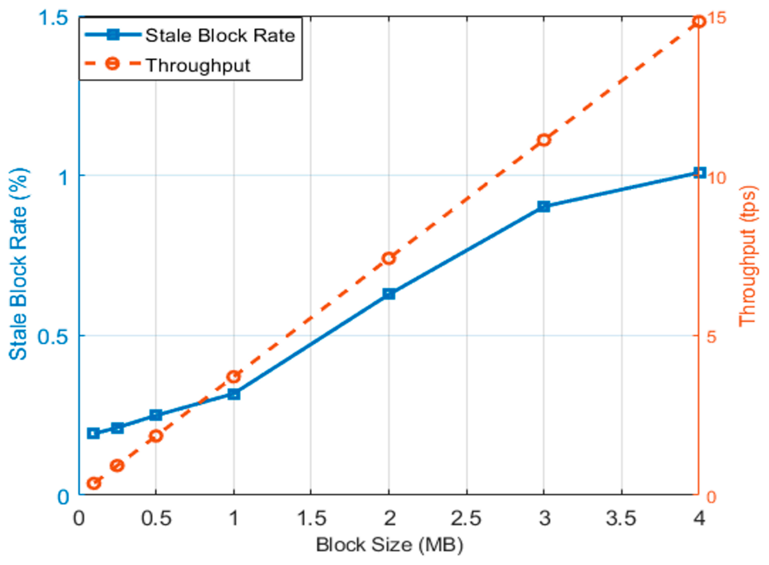

Figure 5. Effect of block size δ on stale block rate and throughput.

Figure 6 shows the results under varying τ obtained for a default δ = 2 MB. It shows that as τ increases, the stale block rate decreases, but so does the throughput. Hence, we can make appropriate choices of δ and τ that meet our stale block rate and throughput requirements. Moreover, the latency incurred by the v-FAP controls the lower limit of τ, since the seed node cannot generate a block before all transactions are received from the service nodes. Alternatively, the δ and τ that meet the required stale block rate and throughput can inform an appropriate choice of N that simultaneously meet the client's latency and MPA requirements.

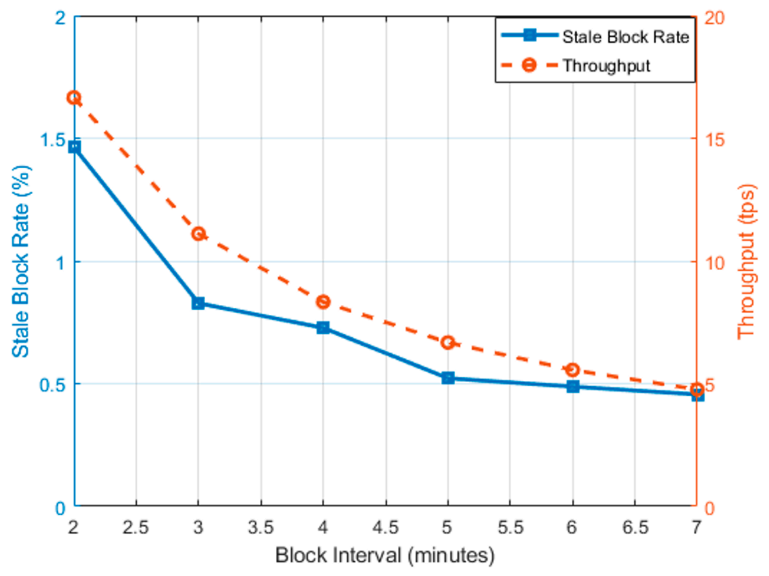

Figure 6. Effect of block interval τ on stale block rate and throughput.

For instance, if the requirements are stale block rate $\leq$ 1%, throughput $\geq$ 10 tps, latency $\leq$ 200 secs, and MPA $\geq$ 92%, then, based on the results in Figure 6, the appropriate settings for our system could be δ = 2 MB; τ = 3 min; and N = 3, for a resulting stale block rate = 0.8%, throughput = 11.5 tps, latency = 161.86 secs, and MPA = 92.19%.

7. Conclusions

For a distributed system such as OF-RAN to be deployed in the real world, it needs a trusted operating environment and efficient means of managing its processes related to v-FAP formation and service execution. This paper seeks to leverage blockchain with smart contracts as an enabler of trusted and distributed systems to propose an automated mechanism using smart contracts for OF-RAN processes. The algorithms defining the logic of the smart contracts are designed, including the service logic for federated DL as a use-case of the proposed mechanism. The resulting smart-contract-enabled OF-RAN is implemented and evaluated through emulation and simulation. The evaluation validates the federated DL performance in terms of latency and precision accuracy under the effect of varying service nodes in a v-FAP. The results show that increasing service nodes reduces the latency, but also marginally decrease the accuracy, suggesting the need to balance a trade-off between the latency and accuracy of training the DL model, according to the client's requirements.

The evaluation also validates the blockchain performance in terms of stale block rate and throughput under the effects of varying block size and block interval. It shows that throughput can be increased by increasing block size, while stale block rate can be decreased by increasing block interval. The appropriate setting of process parameters such as block size, block interval, and number of service nodes to meet the often competing requirements of latency, precision accuracy, stale block rate, and throughput is also demonstrated. As future work, the expected gains in terms of operational efficiency and cost of service delivery for such an automated OF-RAN could be analyzed for different operational scenarios and cost structures.

Author Contributions: Conceptualization, J.J. and B.-C.S.; methodology, J.J. and B.-C.S.; software, J.J.; validation, J.J. and B.-C.S.; formal analysis, J.J. and B.-C.S.; investigation, J.J.; resources, B.-C.S.; data curation, J.J.; writing—original draft preparation, J.J.; writing—review and editing, B.-C.S. and P.H.J.C.; visualization, J.J.; supervision, B.-C.S. and P.H.J.C.; project administration, B.-C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data that support the findings of this study are available from the authors upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Habibi, M.A.; Nasimi, M.; Han, B.; Schotten, H.D. A comprehensive survey of RAN architectures toward 5G mobile communication system. *IEEE Access* **2019**, *7*, 70371–70421. [CrossRef]
2. Peng, M.; Yan, S.; Zhang, K.; Wang, C. Fog computing-based radio access networks: Issues and challenges. *IEEE Netw. Mag.* **2016**, *30*, 46–53. [CrossRef]
3. Jijin, J.; Seet, B.-C. Opportunistic fog computing for 5G radio access networks: A position paper. In Proceedings of the Third International Conference on Smart Grid and Innovative Frontiers in Telecommunications (SmartGIFT), Auckland, New Zealand, 23–24 April 2018.
4. Lilien, L.; Gupta, A.; Kamal, Z.; Yang, Z. Opportunistic resource utilization networks—a new paradigm for specialized ad hoc networks. *Comput. Electr. Eng.* **2010**, *36*, 328–340. [CrossRef]
5. Jijin, J.; Seet, B.-C.; Chong, P.H.J. Blockchain enabled opportunistic fog-based radio access network: A position paper. In Proceedings of the 29th International Telecommunication Networks and Applications Conference (ITNAC), Auckland, New Zealand, 27–29 November 2019.
6. Rathore, S.; Pan, Y.; Park, J.H. BlockDeepNet: A Blockchain-based secure deep learning for IoT network. *Sustainability* **2019**, *11*, 3974. [CrossRef]
7. Cui, L.; Su, X.; Ming, Z.; Chen, Z.; Yang, S.; Zhou, Y.; Xiao, W. CREAT: Blockchain-assisted compression algorithm of federated learning for content caching in edge computing. *IEEE Internet Things J.* **2022**, *9*, 14151–14161. [CrossRef]
8. Rathore, S.; Kwon, B.W.; Park, J.H. BlockSecIoTNet: Blockchain-based decentralized security architecture for IoT network. *J. Netw. Comput. Appl.* **2019**, *143*, 167–177. [CrossRef]
9. Zhu, H.; Huang, C.; Zhou, J. Edgechain: Blockchain-based multi-vendor mobile edge application placement. In Proceedings of the 4th IEEE Conference on Network Softwarization and Workshops, Montreal, QC, Canada, 25–29 June 2018.
10. Liu, Y.; Wang, K.; Lin, Y.; Xu, W. LightChain: A lightweight Blockchain system for industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3571–3581. [CrossRef]
11. Weng, J.; Zhang, J.; Li, M.; Zhang, Y.; Luo, W. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 2438–2455. [CrossRef]

12. Lu, Y.; Huang, X.; Zhang, K.; Maharjan, S.; Zhang, Y. Low-latency federated learning and Blockchain for edge association in digital twin empowered 6G networks. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5098–5107. [CrossRef]
13. Jijin, J.; Seet, B.-C.; Chong, P.H.J. Multi-objective optimization of task-to-node assignment in opportunistic fog RAN. *Electronics* **2020**, *9*, 474. [CrossRef]
14. Gervais, A.; Karame, G.O.; Wüst, K.; Glykantzis, V.; Ritzdorf, H.; Capkun, S. On the security and performance of proof of work blockchains. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016.
15. Zhao, Z.; Feng, C.; Yang, H.H.; Luo, X. Federated-learning-enabled intelligent fog radio access networks: Fundamental theory, key techniques, and future trends. *IEEE Wirel. Commun. Mag.* **2020**, *27*, 22–28. [CrossRef]
16. The MINST Database. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 5 August 2022).