Blockchain-based Network Slice Broker to Facilitate Factory-as-a-Service

***Abstract*—The novel concept of Factory-as-a-Service (FaaS) allows the agility of adapting the manufacturing process by identifying the industry's supply chain and user requirements. To cater to FaaS, flexibility in networking and cloud services is a must. 5G network slice broker is a third-party mediator that caters to networking resource demand from clients to the service providers. Thus, this paper introduces a secure blockchain-based network slice broker to facilitate FaaS. The proposed secure network slice broker (SNSB) provides secure, cognitive, and distributed network services for resource allocation and security service level agreement (SSLA) formation with coordination of slice managers and SSLA managers. In SNSB we introduce a federated slice selection algorithm with Stackelberg game model and Reinforcement Learning (RL) algorithm to compute the real-time and the optimal unit price and demand level. We provide an extensive implementation and performance evaluation of SNSB using the Katana slice manager and a custom SSLA manager.**

I. Introduction

The growing need for Industrial Internet of Things (IIoT) will require scaling up or down in different technological aspects and addressing the societal and environmental changes [1]. Factory-as-a-Service (FaaS) allows the agility of adaptation of the manufacturing process by identifying the supply chain and user requirements in IIoT. To enable FaaS with the help of networking and cloud services, it is always essential to have non-interrupting IT and telecommunication services [2], [3], [4]. When an IIoT site is forming as FaaS, it should scale up or scale down the operations against the new engagements with higher flexibility. Similar to the adjustment of other infrastructure and accessibility of different services, networking and cloud services should also be flexible and adapted to the time requirements. Instead of buying these from a single service provider, the operations in FaaS will have higher flexibility to acquire them from an open marketplace that has access to multiple resource providers (RPs).

Instead of conventional network slice creation by one operator, we propose a more democratic way of forming multi-operator network slices based on the consumer requirements to cater to FaaS. An intermediate third-party service is running as a network slice broker to enable this slicing scenario. 5G network slice broker (NSB) allows the dynamic interoperability and resource trading requirements of market players such as infrastructure providers, consumers, and mobile network operators in trading the network and computational resources [5]. Instead of having a centralized network slice broker, offering brokering service as a blockchain-based distributed service will bring higher flexibility and eliminate the single point of failure [6]. To invoke the security services and meet the security levels requested by the consumers, it is necessary to integrate the predefined security service level agreements (SSLAs) [7] with the corresponding resource providers.

In the previous works, the concept of NSB is presented in both centralized and distributed forms as the running prototypes. At the same time, in the state-of-the-art, blockchain is used as a facilitator to FaaS and even blockchain-based NSBs are proposed for other applications. Although the blockchain-based NSBs are proposed for general applications, they lack applicability in a particular use case and the security considerations like the invocation of SSLAs. Moreover, they do not provide any cognitive slice selection mechanisms which can formulate federated network slices based on the resource demand and availability. Therefore, this paper introduces the blockchain-based NSB to create federated network slices in the context of sharing network resources in FaaS for IIoT applications. Our main contribution is to develop the Secure Network Slice Broker (SNSB) that provides distributed and intelligent network services in resource allocation and SSLA formation with the coordination of slice managers and SSLA managers. To the best of our knowledge, this is the very first research article that presents a secure and trustworthy network slice brokering solution for FaaS as a blockchain-based service using game theory and Reinforcement Learning (RL) based selection algorithm together with a real-life implementation in 5G networking environment by integrating Katana network slice manager [8]. In the federated slice selection algorithm of SNSB, we use a Stackelberg game model and RL algorithm to compute the optimal unit price and demand level of each resource category in real-time. The federated network slice is created based on those obtained optimal values used as a quantitative representation for the resource allocation.

The remainder of the paper is organized as follows: Section II discusses the related works. Then, Section III proposes the brokering architecture for the FaaS use case and explains the slice selection algorithm explicitly. Section IV evaluates the slice selection algorithm and presents the prototype implementation. Finally, section VI concludes the paper.

II. Related Work

As stated in [5], in addition to facilitating on-demand resource allocation, NSB performs admission control based on traffic monitoring and forecasting, including mobility, based on a global network view. It configures radio access network (RAN) schedulers to support multi-tenancy use cases. According to 3GPP specifications and the initial design in [5], the 5G Network Slice Broker is co-located at the master operator-network manager (MO-NM), which monitors and controls the shared RAN, and interacts with the sharing operator network manager (SO-NM). Chaer et al. [9] presents how the blockchain leverages the 5G networks with potential opportunities for the 5G networks, including infrastructure crowdsourcing, infrastructure sharing. Nguyen et al. [10] presents a comprehensive survey on the integration of blockchain for 5G and beyond networks.

Backman et al. [11] highlighted the significance of blockchain as an additional trust layer for NSB. Valtanen et al. [12] present an analysis of a blockchain-based slice brokering use case as a resource configuration framework in the perspective of industrial automation. Boubendir et al. [13] proposed a federated operational architecture to share network and IT resources to the consuming stakeholders.

Afraz et al. [14] defined blockchain as a tradable commodity with the parameters such as RAN, computational resources, and storage. Zanzi et al. [15] proposed NSBChain, which is a hierarchical blockchain architecture for network slice brokering. Nour et al. [16] proposed a blockchain-based network slice brokering mechanism with anonymous transactions. Antevski and Bernardos [17] proposed a distributed-ledger-based solution for the federation of 5G network services through smart contracts.

While evolving the above research efforts towards beyond 5G, the most significant improvement is to bring intelligence by learning the network state and user satisfaction to form a suitable network slice. With this requirement, the role of the network slice broker also needs to be reformed as an entity that operates with higher intelligence. The current state-of-the-art brokering architectures do not provide real-time evaluation of the resource availability and their pricing values offered by different resource providers or do not consider any security properties for network slice creation. Therefore, to bring the intelligence for slicing technology in beyond 5G networks, it is highly required to have a trusted and secured slice brokering architecture that provides real-time services based on the current network resource availability.

III. Brokering Architecture for FaaS

The former part of this section describes the role of the network slice broker in a FaaS use case scenario together with the proposed SNSB architecture and its workflow. Then we provide the mathematical explanation for the slice selection algorithm that we consider in the SNSB.

A. Usecase scenario

In this paper, we consider a use case scenario where a blockchain-based network slice broker facilitates to enable Factory-as-a-Service. A blockchain-based NSB is a distributed trading platform to cater to federated network slices as required by each production site (Figure 1).

In our solution, NSB is a distributed service that collects resource requests and security service requirements from each production site and designs the network slice based on the resource availability and ability to provide security services at the resource providers. For that, NSB requires to keep records of resource availability and security services provided by each resource provider. NSB blockchain service should run on each miner located at the production and resource provider sites.

The potential resource providers proposed in the SNSB include MNOs, local 5G operators, and cloud infrastructure providers who are willing to trade the resources for the service-oriented factories that operate as consumers.

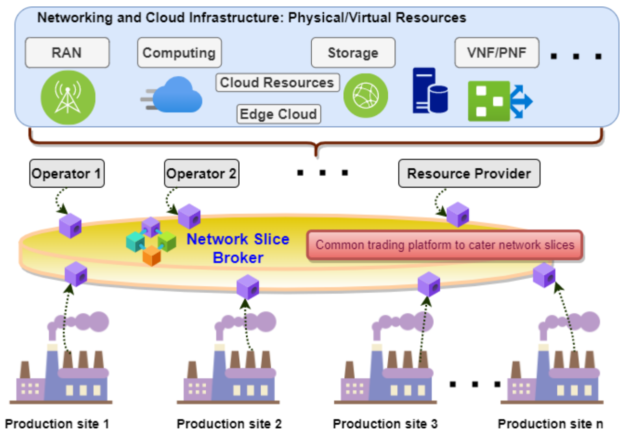


Fig. 1: Role of network slice broker to enable FaaS.

B. Proposed brokering architecture

The SNSB (Secure Network Slice Broker) architecture and its core modules are presented in Figure 2. *Prime Mover* collects the resource requests coming from different production sites and forwards them to *Mediator*. *Security Manager* is a security service blockchain (SSB) to protect the entire brokering service from Denial of Service (DoS) attacks. When the resource requests are coming from the tenants, *Mediator* runs the slice selection algorithm by taking the updates of resource availability and price, creating the Network Slice Template (NST), and sending NST to *Global Slice Manager*. Finally, the *Global Slice Manager* accumulates the slice with the coordination of slice managers and SSLA manager, and send the slice to consumers.

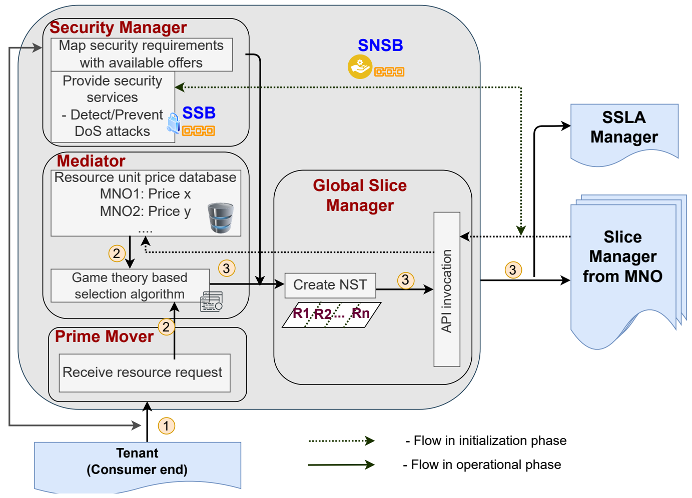

Security Manager
Map security requirements with available offers
Provide security services
- Detect/Prevent DoS attacks
SSB
SNSB
Mediator
Resource unit price database
MNO1: Price x
MNO2: Price y
....
Game theory based selection algorithm
Global Slice Manager
Create NST
R1,R2... Rn
API invocation
SSLA Manager
Slice Manager from MNO
Prime Mover
Receive resource request
Tenant (Consumer end)
1
2
3
- Flow in initialization phase
- Flow in operational phase

Fig. 2: The proposed architecture of SNSB solution

C. Work flow of brokering architecture

The sequence of events executed from the initial stage to the completion stage in our approach is depicted in the figure 3. Initially, a tenant sends a slice request along with their desired security requirements (1) to the proposed framework. Then, the Security Service smart contract verifies the slice request against DOS attacks (2) and sends the confirmed slice request with the security requirements to the SNSB. In the next stage, SNSB runs the slice selection algorithm (4) to find the optimal slice that tallies the tenant and security requirements. The Slice Managers of the selected slice are notified by SNSB

(5). The SNSB acknowledges the SSLA manager that slice has been instantiated and forwards the SSLA information (6). After that, the Slice Manager is responsible for invoking and offering the chosen optimal slice to the tenant (7). Afterwards, the SSLA manager sends the SSLA establishment notification to the tenant (8), based on the tenant's security requirements. Whenever an MNO adds/updates its network resources or the unit prices of the resources, that particular information will be updated by the respective Slice Manager of the MNO. The security service smart contract checks the validity of the updated request received by the resource provider against the DoS attacks, based on profile information stored in the ledger.

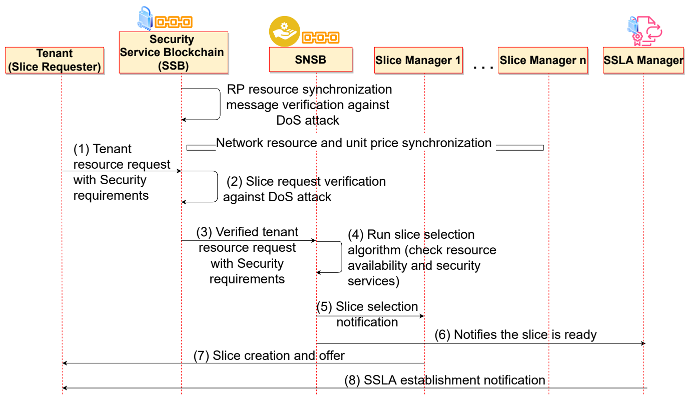

Tenant (Slice Requester)
Security Service Blockchain (SSB)
SNSB
Slice Manager 1
. . .
Slice Manager n
SSLA Manager
RP resource synchronization message verification against DoS attack
Network resource and unit price synchronization
(1) Tenant resource request with Security requirements
(2) Slice request verification against DoS attack
(3) Verified tenant resource request with Security requirements
(4) Run slice selection algorithm (check resource availability and security services)
(5) Slice selection notification
(6) Notifies the slice is ready
(7) Slice creation and offer
(8) SSLA establishment notification

Fig. 3: The workflow of brokering architecture

D. *Slice selection algorithm*

We consider that a particular network slice blueprint is created with n number of resource (or network functions) categories. In a resource request created by a certain fog node, u_i denotes the amount of resource demand for i^{th} resource R_i, where $i \in \{1, 2, ..., n\}$. There are m number of resource providers such that O_j denotes the j^{th} resource provider where $j \epsilon \{1, 2, ..., m\}$. The resource provider (or operator) O_j sets the pricing strategy $\{v_j = [v_{ji} i \in n : 0 < v_{ji} < \overline{v}]\}$ as the unit price of i^{th} resource, where v_{ji} is the price offered and $\overline{v}$ is the maximum price. c is taken as the common and constant cost resulting from the general operation and maintenance cost.

Hence, the expected utility (reward) by O_j resource provider can be presented as:

$$P_j = \sum_{i=1}^{n} u_i v_{ji} - \sum_{i=1}^{n} c u_i \tag{1}$$

In addition to that, we introduce a utility function P_i, which is the expected utility (reward) for R_i resource category requested by the miner node located at Fog Node (based on the offer given by O_j operator). To develop the algorithm, the reward is assigned to each resource category as an indication of its popularity (or demand level) among the consumers:

$$P_i = P \frac{u_i}{\sum\limits_{i=1}^{N} u_i} - v_{ji} u_i \tag{2}$$

The fixed utility P is received as a reward by the miner node for successful mining of a given service request with respect to all the resource categories.

As described above, after receiving all the offers from the resource providers with in a given period of time, the selection algorithm first computes the total service demand of fog nodes and set the offer prices to earn more profit to the resource providers. This optimization problem can be formulated as:

$$\begin{aligned} \max_{v_j} \quad & P_j(v_j|u) \\ \text{s.t.} \quad & y_j \geq 0, \\ & \sum_{i=1}^{N} u_i v_{ji} \geq \sum_{i=1}^{N} c u_i \end{aligned} \tag{3}$$

On the other hand, the miner nodes located in fog nodes, need to maximize the reward received for each resource requirement (category). Therefore, observing the price strategies of resource providers, the selection algorithm will formulate the optimization problem (i.e., to compute the optimum resource demand) of each resource category as:

$$\begin{aligned} \max_{u_i} \quad & P_i(u_i|vji) \\ \text{s.t.} \quad & u_i \geq 0, \\ & P \frac{u_i}{\sum\limits_{i=1}^{N} u_i} \geq v_{ji} u_i \end{aligned} \tag{4}$$

Accordingly, the mathematical model is formulated for two sides in the Stackelberg game. The selection algorithm in SNSB is responsible for updating both the resource providers and the fog nodes about how they are able to continuously modifying strategies to boost their utilities. The aim of the Stackelberg game is to discover the Nash equilibrium, where no player has intention to diverge from its strategy after taking into consideration of its' opponent's selection. In this problem, the Nash equilibrium of the Stackelberg game is modeled as below.

In order to show the feasibility of Stackelberg game, let v^* and u^* be the optimal unit price of one resource provider and the optimal resource demand for each resource category. Therefore, the point (u^*, v^*) is considered the Nash equilibrium point if it satisfies, $P_i(u^*, v^*) \geq P_i(u, v^*)$ and $P_j(u^*, v^*) \geq P_i(u^*, v)$.

Verifying the distinctiveness and presence of the Nash equilibrium in the Stackelberg game is achieved by taking the second order derivatives of utility functions of Eq. 2 and Eq. 1 with respect to u_i and v_j as follows.

$$\frac{\partial^2 P_i}{\partial {u_i}^2} = -2P \frac{\sum\limits_{k \neq i} u_k}{(\sum\limits_{k \in N} u_k)^3} \leq 0 \tag{5}$$

$$\frac{\partial^2 P_j}{\partial {v_j}^2} = -\frac{2P}{{v_j}^2} \frac{(N-1)P}{N} \leq 0 \tag{6}$$

According to the above equations and as described in [6], P_i and P_j utility functions are stringently concave, and the Nash equilibrium present in this Stackelberg game.

In the first part of the selection, the algorithm should be run to fill the values given in Table I. The table is updated for all the available operators, their optimal unit prices as a common indication for all the resource categories, and the optimal demand. This table is taken as a reference to create the federated NST and decide its composition with different operators and resource categories.

TABLE I: Table for optimal unit prices of operators and optimal resource demand from each resource category

		Optimal resource demand				
Operator	**Optimal unit price**	R_1	R_2	R_3	...	R_n
O_1	v_1^*	u_{11}^*	u_{12}^*	u_{13}^*		u_{1n}^*
O_2	v_2^*	u_{21}^*	u_{22}^*	u_{23}^*		u_{2n}^*
...						
O_m	v_m^*	u_{m1}^*	u_{m2}^*	u_{m3}^*		u_{mn}^*

E. Multi-agent Reinforcement Learning algorithm

The Stackelberg game presented above can be solved as a multi-agent reinforcement learning problem, where each player in the game is represented by a learning agent. In multi-agent scenario, each agent aims at maximizing its own cumulative reward, which, in turn, maximizes the total reward of all agents in the system. The key issue of multi-agent reinforcement learning is the non-stationary learning problem due to the effects of actions of other agents. Therefore, reinforcement learning agents often experience oscillatory problem where optimal policy does not converge [18]. For this reason, classical learning algorithms, such as Q-learning, have been modified to achieve better conversion. The Win of Learn Fast Policy Hill Climbing (WoLF-PHC) algorithm is proposed as an extension of the Q-learning algorithm for more efficient learning of the dynamic target [19]. Its characteristic is the use of different learning rates depending on the game outcome, which increases the convergence in multi-agent non-stationary environment. This paper presents the application of WoLF-PHC algorithm to the slice selection problem.

We denote $u_i \in A_i$ and $v_{ji} \in A_j$ the resource R_i demand action of fog node and the unit price set by resource provider O_j for resource i, respectively, where A_i represents the set of all possible actions of the fog node, and A_j set of possible actions of the operator. In each time slot, the miners and the resource provider take actions. At the start of the time slot t, the operator and fog nodes take the actions, i.e., set the price v_{ji}^t and demand u_i^t, based on the observed state of the system. The state of the system for the fog node is described as $s_i = v_{ji}^{t-1}$, which is the observed price for the resource R_i at the previous time slot. The state of the system for the operator is defined as $s_j = [u_i^{t-1}]$, where u_i^{t-1} represents the service demand of each fog node in the previous time slot. The immediate reward is defined by (1) and (2).

Similar to the Q-learning algorithm, the WoLF-PHC algorithm calculates the Q-table. In the multi-agent case, each agent is updating its own Q-table. However, the action selection in the win-of-learn-fast mechanism is different from the classic ϵ-greedy policy, where the action is selected based on the values of the Q-table, i.e., maximum of the actions for the given state in the exploitation phase and random in the exploration phase. WoLF-PHC selects the action based on the policy π. This means that some action a in-state s is selected with probability $\pi[s][a]$. In the rest of the section, we describe the details of the policy updates and the WoLF-PHC algorithm steps on the example of the fog node agent. The operator's agent utilizes the same algorithm for its own set of actions, states, and reward definitions as defined above.

After the all agents have taken actions and observed their rewards, they update the Q-value. We denote the learning rate of the fog node as $\alpha_i \in (0, 1]$ and discount factor as $\gamma_i \in (0, 1]$. The Q-function of the fog node with the service demand u_i in the state s_i^t is updated as:

$$Q_i(s_i^t, u_i^t) = Q(s_i^t, u_i^t) + \alpha_i(P_i + \gamma\, max_{u_i} Q(s_i^{t+1}, A_i) - Q(s_i^t, u_i^t)) \tag{7}$$

The WoLF mechanism, keeps track of the current average policy $\overline{\pi_i}(s_i^t, u_i)$ which is used in order to decide the "win" or "lose" of the policy $\pi_i(s_i^t, u_i)$. The fog node agent selects its learning parameter θ_i from $\theta_{i,win}$ and $\theta_{i,lose}$ where $\theta_{i,win} < \theta_{i,lose}$. If the agent is winning, $\theta_{i,win}$ updates the policy cautiously. Otherwise, $\theta_{i,lose}$ is used to learn fast from the lost game. The policy is the winning policy if the following criteria is met:

$$\sum_{u_i \in A_i} \overline{\pi_i}(s_i^t, u_i) Q_i(s_i^t, u_i) > \sum_{u_i \in A_i} \pi_i(s_i^t, u_i) Q_i(s_i^t, u_i) \tag{8}$$

For the computation of the current average policy, $N_i(s_i^t)$ is used to record the occurrence count of states noticed by the agent, that is increased by one each time the system is in the state s_i^t. Then, the average policy of the fog node can be updated as:

$$\overline{\pi_i}(s_i^t, u_i) = \overline{\pi_i}(s_i^t, u_i) + \frac{\pi_i(s_i^t, u_i) - \overline{\pi_i}(s_i^t, u_i)}{N_i(s_i^t)}\ \forall u_i \in A_i \tag{9}$$

In the course of the learning process, the chance of the fog node selecting a service demand is progressively increased, which can elevate the expected reward, followed by the reduction of the other actions [19]. Hence, the update of the service demand policy of the fog node can be presented as follows,

$$\pi_i(s_i^t, u_i) = \pi_i(s_i^t, u_i) + \Delta\ \forall u_i \in A_i \tag{10}$$

where:

$$\Delta = \begin{cases} -min(\pi_i(s_i^t, u_i), \frac{\theta_i}{|A_i|}), & \Pi \\ \sum_{u_i \neq u_i'} min(\pi_i(s_i^t, u_i), \frac{\theta_i}{|A_i|}), & \text{otherwise} \end{cases} \tag{11}$$

where $\mid A_i \mid$ is the number of actions and condition Π is:

$$\Pi u_i \neq argmax_{u_i' \in A_i} Q_i(s_i^t, u_i') \tag{12}$$

The WoLF-PHC algorithm steps for the fog agent are summarized in Algorithm 1.

Algorithm 1 The WoLF-PHC algorithm for the fog node

```
1: Set $\alpha_i$,$\delta_i$,$\theta_{i,win}$,$\theta_{i,lose}$
2: Initialize $t = 0$, $Q_i(s, u) = 0$,$\pi_i(s, u) = 1/ \mid A_i \mid$ and $\overline{\pi_i}(s, u) = 1/ \mid A_i \mid \forall s, u$
3: Repeat
4: Observe environment state $s_i^t$
5: Select action $u_i^t$ at random with the probability policy $\pi_i(s_i^t, u_i)$
6: Observe next state $s_i^{t+1}$ and immediate reward $P_i$
7: Update $Q_i(s_i^t, u_i^t)$ according to (7)
8: Update $\pi_i(s_i^t, u_i)$ according to (10)
9: $t = t + 1$
10: until
```

In the course of the training process, the agent updates its strategy based on (7). Hence, the complexity of the training process of each agent is in the order of $O(S^2A)$, where S stands for the size of the state space and A stands for the size of the action space. As for the complexity of the running process of WoLF-PHC, considerably complex is the Q-table look-up. Hence, the complexity of the running process of each agent is approximately $O(S)$.

F. Deployment of smart contracts

The proposed architecture leverages the slice brokering operation by encoding the selection algorithm into smart contract in the consortium blockchain. The consortium comprises of factories and resource providers. As indicated in Figure 2, the Tenants slice managers and SSLA managers connected using Application Programming Interfaces (APIs). The slice brokering service is deployed in the local blockchain nodes with the APIs to invoke slice request($Step1$) from tenant end as indicated in Figure 2. Inward transactions APIs are encoded in the $PrimeMover$ smart contract. Outward API invocations ($Step3$) are encoded in the $Mediator$ smart contract. There

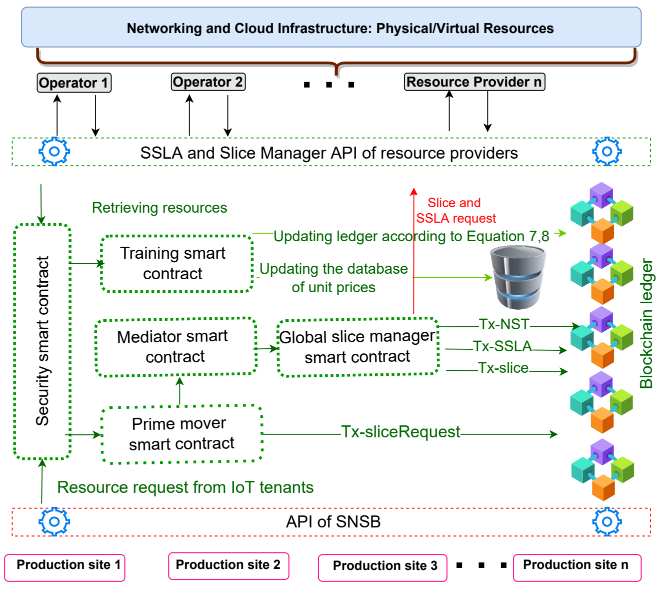

Networking and Cloud Infrastructure: Physical/Virtual Resources
Operator 1
Operator 2
Resource Provider n
SSLA and Slice Manager API of resource providers
Retrieving resources
Slice and SSLA request
Updating ledger according to Equation 7,8
Training smart contract
Updating the database of unit prices
Mediator smart contract
Global slice manager smart contract
Tx-NST
Tx-SSLA
Tx-slice
Security smart contract
Prime mover smart contract
Tx-sliceRequest
Blockchain ledger
Resource request from IoT tenants
API of SNSB
Production site 1
Production site 2
Production site 3
Production site n

Fig. 4: Smart contract interaction

are five different smart contracts that synergistically operate to ensure the end to end operation of the slice brokering process. The interaction between each smart contract illustrated in Figure 4. The descriptions of the role of each smart contract are as follows.

*1) **Training smart contract**:* Training smart contract performs the training operation to formulate the Q values and service demand policies. Algorithm 1 is encoded in the training smart contract and updates of $Q_i(s_i^t, u_i^t)$ according to (7) and update $\pi_i(s_i^t, u_i)$ according to (10) performed by the training smart contract. The corresponding values are stored in the ledger. The training smart contract operates as an offline smart contract without real-time slice brokering. The training smart contract executes in a timely execution interval for the synchronization of the training data set.

*2) **Prime Mover smart contract**:* This smart contract hosts the inward API received from the consumer end. The inward transaction $Tx_{sliceRequest}$ received by the tenant T_α can be defined as,

$$Tx_{sliceRequest} =< T_\alpha, R_\alpha, Timestamp > \quad (13)$$

where the resource request R_α

$$R_\alpha = \{u_i \dots u_n\} \quad (14)$$

The ledger stores the data elements received from $Tx_{sliceRequest}$ in the ledger to ensure non-repudiation of the tenants regarding the resource request.

*3) **Mediator smart contract**:* This smart contract retrieves the available resources and unit prices from the resource unit price database for the computation of optimal resource providers to deliver the optimal resource for the consumers. According to Figure 2, the game theory-based selection algorithm (Algorithm 1) executes in the Mediator smart contract.

*4) **Global slice manager**:* The global slice manager smart contract creates the network slice template of the federated slice. Global slice manager invokes the resource provider slice managers and SSLA managers to instantiate the slice. The transactions committed to the ledger in global slice manager include Tx_{NST} which can be defined as,

$$Tx_{NST} =< T_\alpha, \{u_i * w_j \dots u_n * w_m\}, Timestamp > \quad (15)$$

where w_j is the quantity of resources provided from the RP j (j=1 to j=m). The transaction committed into the ledger ensures the integrity of formulated slice offer from the SNSB. Furthermore, the SSLA transaction is defined as,

$$Tx_{SSLA} =< T_\alpha, SSLA-IDentifier, Timestamp > \quad (16)$$

and the slice instance transaction can be defined as,

$$Tx_{slice} =< T_\alpha, Slice-IDentifier, Timestamp > \quad (17)$$

*5) **Security manager smart contract**:* The security manager smart contract performs the security verification of inward transactions. Especially, the tenant resource requests which exceed the maximum authorized request frequency and authorized quantity limits will be blocked.

G. Deployment of blockchain nodes

SNSB proposed to facilitate the slice brokering as a service to the intervening stakeholders of slice brokering operation in the manufacturing process. Decentralizing the slice brokering service towards the stakeholders is one of the key requirements in SNSB. As indicated in Figure 1, the key categories of stakeholders are twofold. The MNOs, local 5G operators, and cloud service providers can be broadly categorized as resource providers while the manufacturing plants which consume the 5G resources in the manufacturing processes can be categorized as consumers. Each stakeholder, either resource provider or consumer requires to onboard to the slice brokering ecosystem. After an agreement between the members of the blockchain network, the stakeholder instantiates a blockchain node locally with the established connectivity to the consortium. The APIs of SNSB instantiates in parallel with the instantiation of smart contracts in the blockchain node.

If the new member is a consumer who anticipates network slice brokering for the manufacturing process, the member can initiate inward resource requests to the new blockchain node to invoke the SNSB for demanding federated slices to advance the manufacturing operation. In contrast, if the new member is a resource provider, the access can be granted to the local

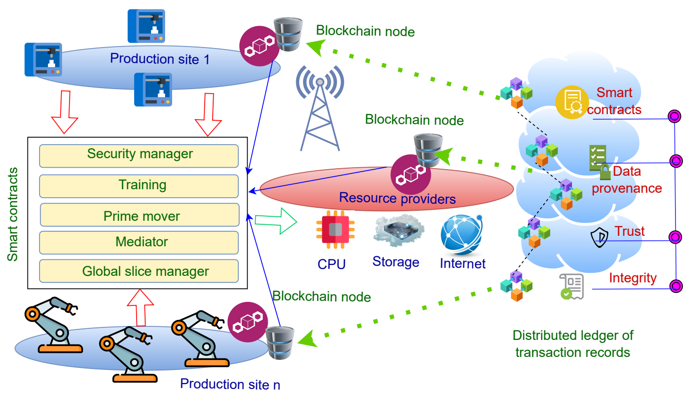


Fig. 5: Blockchain deployment architecture

services such as slice manager and SSLA manager expose the available resources to formulate the federated slice(s) for upcoming resource requests from the consumers. Each stakeholder should contribute to the mining process and the blocks mined within the intermediary steps of slice brokering requires approval of each stakeholder to fulfil the consensus. Figure 5 reflects the blockchain deployment model of SNSB to facilitate FaaS.

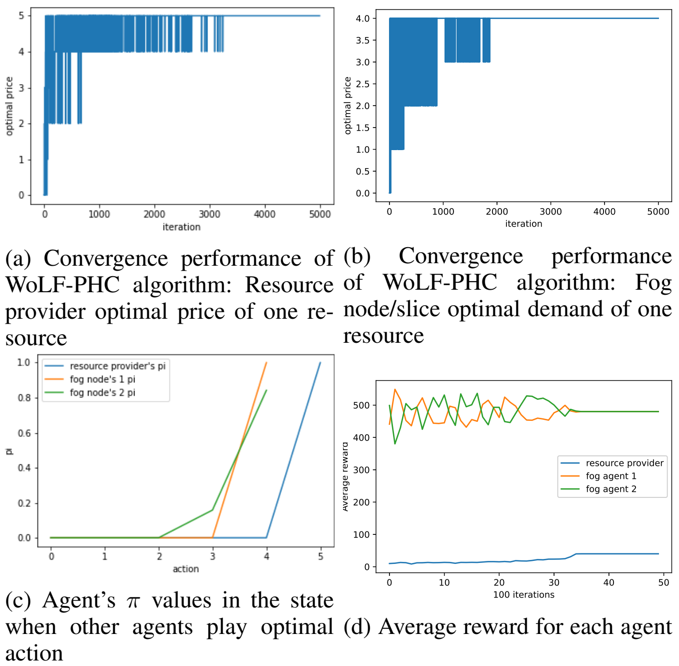
Technically, a similar implementation can be performed using a public blockchain. However, the limitations of the public blockchains such as higher block mining time, the overhead of growing ledger and higher computational overhead for block mining deviate from the proposed architecture from the public blockchain. The heavy computations such as WOLF-PHC algorithm execution and slice selection will incur execution cost (gas cost) in public blockchain such as Ethereum. In contrast, the consortium blockchain provides more flexibility in block mining time and comparably less overhead in the ledger storage which makes the proposed architecture mostly suited for consortium type blockchain implementation.

IV. Performance Evaluation

This section provides the evaluation of the slice selection algorithm and slice brokering architecture. First, the slice selection algorithm is developed in Python. We evaluated a simple example with two fog node agents and one service provider agent as proof of the concept. The proposed solution can be generalized to any number of fog nodes and service providers. Later, the SNSB blockchain service is implemented on Hyperledger Fabric by integrating a real network slice manager and the SSLA manager.

A. *Evaluation of slice selection algorithm*

Initially, we assess the convergence of the WoLF-PHC algorithm. For clarity, let the pricing action set of the resource provider and resource demand action set of the fog node be $A_j = (0, 1, ..., 5)$ and $A_i = (0, 1, ..., 4)$, respectively. The quantitative factor of the cost of the unit resource in operator is $c = 1$, and the reward is $P = 10$. For simplicity, we made an assumption of a uniform pricing strategy for each fog node. To guarantee that the agent can converge to the optimal policy, we set the maximum episode numbers as 5000. Additionally, the learning rate is $\alpha = 0.2$, which decides the degree to which the altered Q-value overrides the previous one. The discount factor



(a) Convergence performance of WoLF-PHC algorithm: Resource provider optimal price of one resource

(b) Convergence performance of WoLF-PHC algorithm: Fog node/slice optimal demand of one resource

(c) Agent's π values in the state when other agents play optimal action

(d) Average reward for each agent

Fig. 6: Evaluation results of slice selection algorithm

$\gamma = 0.8$ expresses how much we focus on future rewards. The learning parameters $\theta_{i,win}$ and $\theta_{i,lose}$ are set to 0.001 and 0.0025, respectively.

The WoLF-PHC algorithm exhibits fast convergence performance owing to the automatic alteration of the learning rate, as depicted in Fig. 6a and 6b, where the optimal resource price and demand are plotted for each iteration of the simulation. Both the resource provider and the fog node agent converge close to the Nash equilibrium point-earning advantages from the "winning or learning fast" mechanism. This exhibits a good trend for selecting the optimal price based on the Nash equilibrium point.

In Fig. 6c, the values of the π functions are plotted for each agent after the system converges. In the figure π, values are given for the system's state when all the agents perform optimal actions in the previous instant. From the figure, it can be seen that each agent has one action that is better than the others, which is easily picked as the optimal (or the best) action to perform.

Fig. 6d shows the average reward of each agent throughout the learning iterations. The reward is averaged over 100 iterations per data point (i.e., 5000). The resource provider's reward is increasing as the agent is converging towards optimal policy. On the other hand, the rewards of the fog nodes depend on the winner of the game. Once the agents converge to the optimal policy, it can be noticed that they obtain equal rewards by assuring the system is fair.

B. *Evaluation of SNSB blockchain service*

Moreover, the discussion of the cloud The experimental setup was developed to evaluate the proposed architecture in a near realistic environment. We implemented SNSB selection algorithm instances of the Hyperledger Fabric blockchain network. Furthermore, virtual infrastructure is simulated using Devstack, which is the developer version of OpenStack platform. Katana slice manager has been integrated with SNSB to

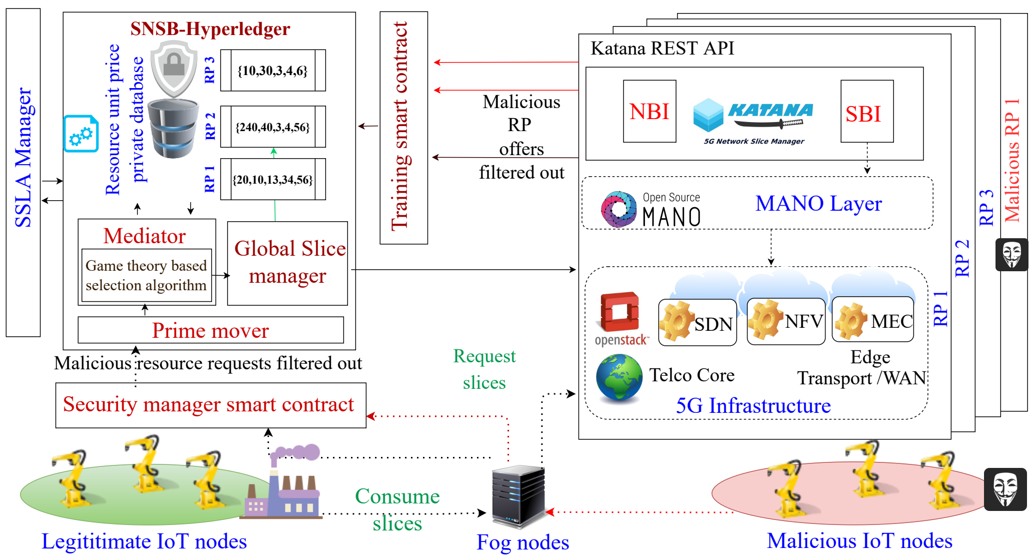


Fig. 7: Testbed implementation setup

invoke slice instantiation requests upon selecting the optimal slice. Implementation setup of SNSB illustrated in Figure 7.

Blockchain service implementation: In the implementation setup, the five smart contracts indicated in Figure 4 are implemented on the Hyperledger Fabric blockchain platform using Java programming language. MQTT and REST APIs are used in the integration for the integration of each service.

Resource unit price database: The resource unit price storage has been implemented on MongoDB database. The MNOs can access the MongoDB storage for updating pricing information. The mediator smart contract accesses the updated pricing information to select the optimal slice based on the consumer request.

RP infrastructure setup: Each RP deployed with the instances of Katana slice manager, Open MANO, and OpenStack for the near-realistic infrastructure integration. Katana slice manager is integrated with REST API for slice creation.

NSBChain implementation for the comparison with SNSB: We have performed a comparison on resource provider utilization and federated network slice pricing with SNSB and NSBChain. We selected NSBChain as it was developed using blockchain and mostly related work to our proposal.

C. Latency evaluation on slice selection

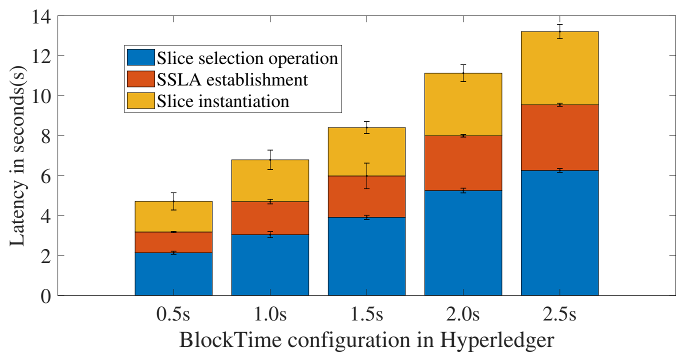


Fig. 8: End-to-end slice creation latency

We have implemented the system to evaluate the latency on the proposed architecture's end-to-end operation. The evaluation has distinguished key sub-operations as slice selection, SSLA establishment, and slice instantiation.

The tenant end is implemented with a software program to simulate the consumer resource request scenario. The SSB and SNSB, encoded in the Hyperledger blockchain platform, have been integrated through APIs with Katana slice manager. SNSB executes the sequential steps within the end-to-end operation upon reaching the consensus within the blockchain network. Block mining time defines the time interval to generate a block that the consensus procedure has approved. The configuration is defined as $BlockTime$ in the Hyperledger Fabric blockchain platform.

Figure 8 reflects the end-to-end slice creation latency in different block mining time configurations in the SNSB blockchain. We programmatically simulated the resource request generation from the IoT tenant end. The experiment includes the end-to-end slice creation process from $Step1$ to $Step8$, which is indicated in 3. Since we need to evaluate the performance impact of SNSB, we fixed the SSB block mining interval($BlockTime$) and changed the SNSB block mining interval for different values.

In this experiment, the software program generated a resource request and end-to-end latency measured on each trial for a specific $BlockTime$ configuration. We performed 100 trials for each $BlockTime$ configuration and measured the latency on slice selection(brokering) operation, SSLA establishment, and selected federated slice instantiation.

Figure 8 reflects the end-to-end slice creation latency in different block mining time configurations in the blockchain. The results show that the impact on the $BlockTime$ is significantly higher for the slice selection than the SSLA establishment and slice instantiation. The slice brokering operation consists of more block mining steps in the proposed architecture, including resource request validation, resource offer validation, and selection result validation. Hence, the block mining impact of $BlockTime$ is higher in the slice selection operation. Furthermore, SSLA operation and slice instantiation operation include also include block mining operations to ensure non-repudiation by maintaining operation status as blockchain transaction logs. Furthermore, ledger records in the interactions with external parties such as SSLA services and Katana slice manager insightful evidence in case of a dispute resolution.

D. RP utilization comparison in SNSB vs NSBChain [15]

SNSB facilitates the federated slice creation based on the optimal resource demand and optimal unit price determination by the smart contract. The smart contract utilizes the Table I in the creation of slice by federating multiple resource providers. We programmatically generated the optimal values and optimal resource demands based on the RL algorithm (Table I)and utilized them for the experimental evaluation. We also defined the fixed number of resource providers (m) into 100 and scaled up the number of different resource types from 10 to 100. The consumer resource request simulated by programmatic generation on fixed-ranged($30-100$) random values for each resource parameter (u_1 to u_n) quantity in the experiment. The corresponding resource offers are generated for all 100 resource providers and stored in the MongoDB database. The same resource request and set of resources offer input to NSBChain [15] and SNSB. For each offer, NSBChain selects the network slice, considering the lowest offer provided by the resource providers. In contrast, SNSB provides a **federated** network slice, which has been formulated based on the optimal unit prices and optimal resource demands indicated in Table I. In this experiment, we have evaluated the resource provider utilization percentage comparison with SNSB and NSBChain. We generated 100 requests per trial for each resource quantity(N) configuration and calculated each trial's mean and standard deviation.

Figure 9 reflects the resource provider utilization percentage on federated slice creation vs NSBChain. The NSBChain always utilizes one resource provider with the lowest price slicing offer, regardless of the number of resources in the tenant request. Hence, the resource utilization remained 1% for the NSBChain within the entire experiment. In contrast, SNSB utilizes multiple resource providers, based on the criteria of Table I and formulates the federated slice. The resource provider utilization increases approximately up to 70 % when the number of resources (N) has increased in the slice formulation. The federated slice ensures the delivery of services which is composed of optimal individual resources according to Table I with competitive pricing. According to the results, SNSB is ideally compatible with multi-operator/resource provider slice brokering scenarios for future industrial use cases with a better utilization percentage than NSBChain.

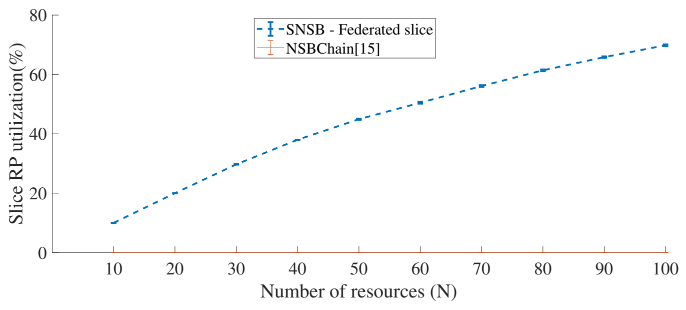


Fig. 9: Resource provider utilization - NSBChain vs SNSB

E. Consumer pricing comparison in SNSB vs NSBChain [15]

SNSB delivers the federated slice for a specific resource request from the optimal unit prices in multiple resource providers. SNSB determines the composition of different resources from each resource provider based on the Table I. We have programmatically generated the resource requests and offers to simulate realistic consumer demand scenarios. A federated slice created by the SNSB is composed of the optimal price offers of different resource providers. We define the finalized price of the federated slice k as,

$$Price_k = \sum_{i=1}^{N} u_i v_j^f \tag{18}$$

where v_j^f is the SNSB selected unit price of federated slice for the parameter j. The $Price_k$ is the summation of all unit prices multiplied by the demand quantities u_i where $i=1$ to $i=n$. We have defined the final $Price_k$ to represent in a generic pricing unit U, making the comparison clearer. In this experiment, we evaluated the behaviour of the mean $Price_k$ when the number of resources (N) remained fixed and the number of resource providers was increased. The NSBchain selects the lowest price-providing resource provider based on the unit prices multiplied by the demand quantities of each resource provider as a single entity. In contrast, SNSB considers the optimal unit prices and optimal resource demands in Table I to offer the federated slice from different resource providers. In the simulation, we provided identical resource requests and offers to the NSBChain and SNSB to compare the offer pricing determined by different algorithms. We performed 100 trials ($k=1$ to $k=100$) for each resource provider quantity setting($10, 20, 30...$ to 100). The mean and standard deviation of the output prices are calculated in different scenarios. Figure 10 reflects the behaviour of mean prices obtained by NSBChain and SNSB.

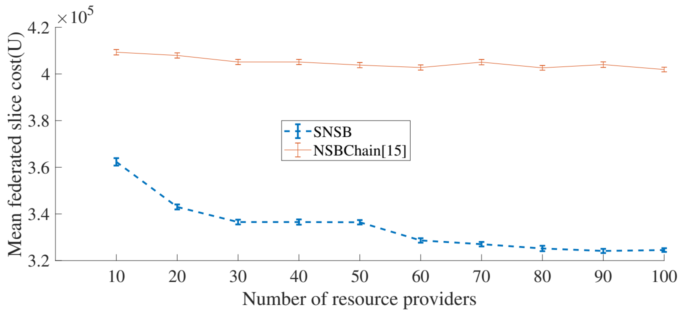
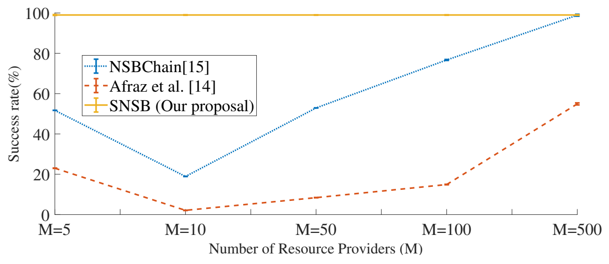


Fig. 10: Offer price comparison - NSBChain vs SNSB

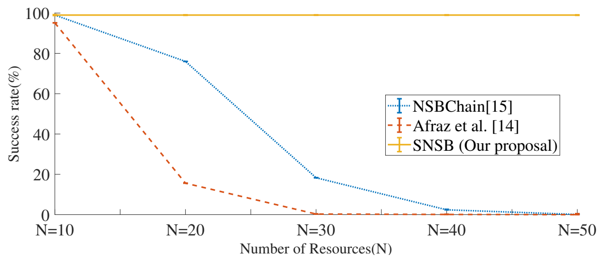
From the results, it is obvious that SNSB yields better performance in consumer price with the delivery of federated slices. Technically, the federated slice is composed of the selected optimal prices from each resource provider. Therefore, the federated slice provides better prices from the consumer's perspective. The mean prices offered by the SNSB decline when the number of resource providers increased as per the Figure 10. The main reason for price decline is the availability of increased options for slice federation when the number of resource providers has been increased. According to the results, it is obvious that SNSB outperforms NSBChain by opening the competitive slice brokering capability with better pricing options for the consumers. The increased number of resource providers can be anticipated in future industrial use cases which leverage the connectivity with local 5G operators.

F. Candidate slice success rate of SNSB, [14] and [15]

Generally, the resource providers deliver the network slice which corresponds to the consumer resource request based on the conditions including the resource availability of requested quantity and pricing compliances. For example, in auction-based slice brokering use cases, the resource provider does not trade if the minimum price for the unit has not been proposed by any of the consumers. In addition to that, in non-federated slice scenarios, the resource providers offer the slice as a single commodity bundled with individual resource unit availability quantities. If any of the resource parameters do not comply with the demand quantity specified in the resource request for that parameter, the offer is invalid as the proposed slice will not be accepted by the consumer. In this experiment, we have defined the success rate as the percentage of possible candidate slice offers from the resource providers' offers to perform slice selection. We have implemented [14] and [15] and input the similar input of resource request and resource provider offers to three algorithms, including SNSB. We randomly generated the number of resource requests and available resource volumes. For the evaluation of [14], which is based on the original algorithm defined in [20], we programmatically generated the appropriate ask\bid values on each resource request scenario. We calculated the percentage of successful slice offers, which can be used as a candidate set to perform a selection of the optimal slice.



(a) Success rate on scaling up the number of resource providers



(b) Success rate on scaling up the number of resources

Fig. 11: Success rate comparison results

Figure 11 reflects the comparison results of the evaluation. Figure 11a indicates the success rate of three algorithms in scaling the number of resource providers(M) while maintaining the number of resources (N) as a fixed value within the entire experiment. According to the algorithm, NSBChain [15] and the work of Afraz et al. [14] indicates a fluctuation in the success rate when the number of resource providers increased. When the number of resource providers increased, the number of single commodity type resource offers getting increased and the candidate slices also increased in parallel [15] and [14]. In contrast, the federate slice approach of SNSB maintains a persistently higher success rate in lower as well as higher resource provider scenarios. Furthermore, the valid candidate single commodity type slices are filtered out further in [14] with the non-compliance of ask\bid, which will eventually reduce the number of valid candidates slice offers. This evaluation simulates the widening of resource providers beyond the MNOs towards local 5G operators.

Figure 11b indicate the successful slice candidate percentages of three algorithms when the number of resources(N) is increased while maintaining the number of resource providers (N) as a fixed value. The experiment reflects a drastic drop of successful candidates in [15] and [14]. The core reason for such an observation is the limitations of delivering a single commodity slice that fulfils all the demanding quantity requirements and compliance ask\bid requirements in [14]. In contrast, the federated slice approach of SNSB maintains a higher success rate in resource scaled up scenarios.

From the results, the federated slice approach proposed in SNSB outperforms the key related work [15] and [14] in terms of successful candidate slice percentage. The key reason for the distinguishing success rate of SNSB is the feasibility to compose the slice with multiple resource providers to deliver the consumer request.

V. Discussion

A. Position of the proposed solution with the state of art

TABLE II: Features comparison with key related works

Features	[5]	[13]	[14]	[15]	[16]	[17]	SNSB
Blockchain-based decentralization	✗	✗	✗	✓	✓	✓	✓
Optimal slice selection through federation of multiple resource providers	✗	✗	✗	✓	✓	✓	✓
Request validation against DoS attacks	✗	✗	✗	✗	✗	✗	✓
Automated SSLA establishment for the federated slice	✗	✗	✗	✗	✗	✗	✓
Experimental evaluation	✗	✗	✓	✓	✓	✓	✓

Table II summarizes the limitations of the state of the art and feature-wise comparison with key related works. We considered the number of citations and implementation feasibility, technical detail availability(pseudocode algorithms) as the selection criteria. The resource requests verified against DoS attacks in the proposal to ensure that the SNSB operation for the legitimate resource requests persists with malicious resource requests. From the numerical result analysis, the proposed algorithms yield optimum mean federated slice cost when compared with the state of the art using the game theory-based selection algorithm. In the success rate oriented experimental evaluations, SNSB yields a higher success rate when compared with the state of art in both scenarios of scaling up the resource providers (M) and scaling up the resources(N). The higher success rate provides a broader selection scope for the slice broker to extend the benefits towards consumer with more competitiveness. Overall, the proposed architecture outperforms the state of the art in feature-wise and numerical figures for different conditions.

B. Implementation challenges and limitations

We propose a blockchain-based solution that can be implemented using either public or consortium blockchain. The implementation challenges in the proposed architecture were originally linked with the blockchain incorporation. Few significant implementation challenges are summarized as follows.

1) ***Block mining latency****:* This is one of the challenges as indicated in Figure 8. The proposed architecture has to align with the blockchain principles. Therefore, the transaction execution latency includes sequential block mining latency. However, we proposed to use a consortium blockchain, which provisions the consortium members to adjust the block mining interval. Providing powerful infrastructure as well as adjusting the block mining interval to align with throughput reduces the impact of block mining latency.

2) ***Computational overhead****:* The block mining requires a set of cryptographic operations to be performed in the transaction validation and the consensus process. The computations in signature generation and verification incurs an overhead on the fog computational infrastructure. For the data provenance of the intermediary transactions in the slice brokering process, the computational overhead requires to be accepted. Applying more lightweight digital signature mechanisms reduce the computational overhead in digital signatures.

3) ***Ledger storage overhead****:* The ledger size expands with the expansion of transactions. Each member node requires to keep a copy of the ledger and the increasing ledger size incurs a storage overhead to the computing infrastructure. A storage recycling and purging mechanism will eliminate the ledger expansion storage overhead.

VI. Conclusions and Future Work

In this paper, we presented the SNSB slice brokering architecture to facilitate FaaS, considering a multi-operator multi-tenant scenario. SNSB is a blockchain-based distributed service that creates federated network slices using a slice selection algorithm and offers secure network slices adhering to the predefined SSLAs. The slice selection algorithm is developed as a Stackelberg game that takes updates from an RL algorithm. The results of the RL algorithm's performance in the simulated network slicing scenario show that fog node agents and resource provider agents can efficiently solve optimization problems to find the optimal price and demand of the resource. Since the complexity of the WoLF-PHC algorithm is low, the slice broker has good scalability as the number of players increases. In SNSB, we encoded the slice selection algorithm as smart contracts to enable decentralized operational capability and ensured non-repudiation. The implementation and evaluation results prove that SNSB outperforms state-of-the-art blockchain solutions in terms of resource provider utilization, lower-priced federated slice formulation, as well as increased success rate for slice selection. We have also pointed the limitations and challenges of proposed work. Finally, we distinguished the advancement of our work in a feature-wise comparison which will ideally fit our work for future industrial application scenarios.

The future work focuses to investigate the potential to eradicate the limitations identified. The storage scalability is one of the features to be improved in the proposed architecture. Moreover, the privacy of transaction data which also provides verifiability in dispute resolution is a potential future work.

References

[1] A. H. Sodhro, S. Pirbhulal, and V. H. C. De Albuquerque, "Artificial intelligence-driven mechanism for edge computing-based industrial applications," *IEEE Transactions on Industrial Informatics*, 2019.

[2] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6g networks," *IEEE Transactions on Industrial Informatics*, 2020.

[3] K. Kaur, S. Guo, M. Chen, and D. Rawat, "Transfer learning for 5g-aided industrial internet of things," *IEEE Transactions on Industrial Informatics*, 2021.

[4] N. Kumar, S. Aggarwal, and P. Raj, *The Blockchain Technology for Secure and Smart Applications across Industry Verticals*. Academic Press, 2021.

[5] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From Network Sharing to Multi-tenancy: The 5G Network Slice Broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[6] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based industrial internet of things," *IEEE Transactions on Industrial Informatics*, 2019.

[7] C.-Y. Lee, K. M. Kavi, R. A. Paul, and M. Gomathisankaran, "Ontology of secure service level agreement," in *2015 IEEE 16th International Symposium on High Assurance Systems Engineering*. IEEE, 2015.

[8] "Katana Slice Manager," https://github.com/medianetlab/katana-slice-manager/wiki, Accessed on 28.12.2021.

[9] A. Chaer, K. Salah, C. Lima, P. P. Ray, and T. Sheltami, "Blockchain for 5G: Opportunities and Challenges," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[10] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5G and beyond Networks: A State of the Art Survey," *Journal of Network and Computer Applications*, p. 102693, 2020.

[11] J. Backman, S. Yrjölä, K. Valtanen, and O. Mämmelä, "Blockchain Network Slice Broker in 5G: Slice Leasing in Factory of the Future Use Case," in *2017 Internet of Things Business Models, Users, and Networks*. IEEE, 2017, pp. 1–8.

[12] K. Valtanen, J. Backman, and S. Yrjölä, "Creating Value through Blockchain Powered Resource Configurations: Analysis of 5G Network Slice Brokering Case," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2018.

[13] A. Boubendir, F. Guillemin, C. Le Toquin, M.-L. Alberi-Morel, F. Faucheux, S. Kerboeuf, J.-L. Lafragette, and B. Orlandi, "Federation of Cross-domain Edge Resources: A Brokering Architecture for Network Slicing," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018.

[14] N. Afraz and M. Ruffini, "5g Network Slice Brokering: A Distributed Blockchain-based Market."

[15] L. Zanzi, A. Albanese, V. Sciancalepore, and X. Costa-Pérez, "NS-Bchain: A Secure Blockchain Framework for Network Slicing Brokerage," *arXiv preprint arXiv:2003.07748*, 2020.

[16] B. Nour, A. Ksentini, N. Herbaut, P. A. Frangoudis, and H. Moungla, "A Blockchain-based Network Slice Broker for 5G Services," *IEEE Networking Letters*, vol. 1, no. 3, pp. 99–102, 2019.

[17] K. Antevski and C. J. Bernardos, "Federation of 5G services using Distributed Ledger Technologies," *Internet Technology Letters*, 2016.

[18] L. Busoniu, R. Babuska, and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2008.

[19] K.-S. Hwang, C.-J. Lin, C.-J. Wu, and C.-Y. Lo, "Cooperation Between Multiple Agents Based on Partially Sharing Policy," vol. 4681, 2007.

[20] N. Afraz and M. Ruffini, "A sharing platform for multi-tenant PONs," *Journal of Lightwave Technology*, vol. 36, no. 23, pp. 5413–5423, 2018.