# Network Traffic Prediction Based on Blockchain and Decentralized Federated Learning with Node Incentive Mechanism in NFV Environment

Ying Hu , Ben Liu , Jianyong Li * , Linlin Jia

Posted Date: 2 May 2025

doi: 10.20944/preprints202504.2606.v1

Keywords: Network Function Virtualization (NFV); Federated learning (FL); Blockchain; Network traffic prediction; Virtual network function migration (VNF migration); Dynamic role switching

 doi:10.20944/preprints202504.2606.v1



*Article*

# Network Traffic Prediction Based on Blockchain and Decentralized Federated Learning with Node Incentive Mechanism in NFV Environment

**Ying Hu, Ben Liu, Jianyong Li * and Linlin Jia**

School of Computer Science and Technology, Zhengzhou University of Light Industry, China
* Correspondence: lijianyong@zzuli.edu.cn

**Abstract:** In the Network Function Virtualization (NFV) environment, dynamic network traffic prediction is crucial for efficient resource orchestration and service chain optimization. When network service providers and resource providers collaborate to deliver services, traditional centralized prediction models face risks such as single points of failure, data privacy leaks across merchants, and malicious node attacks. To address these issues, we propose a blockchain based decentralized federated learning method for network traffic prediction, incorporating a node incentive mechanism. By dynamically adjusting node roles (e.g., "Aggregator," "Worker," " Residual node," and "Evaluator") based on prediction accuracy and network migration performance, the framework effectively mitigates the impact of malicious nodes. Additionally, the decentralized federated learning architecture eliminates reliance on central servers, enhancing collaborative training efficiency among multiple service providers while ensuring data privacy. Simulation experiments conducted on an NFV platform using real-world traffic datasets demonstrate that the proposed method improves prediction accuracy, reduces VNF migration overhead, and maintains prediction stability even in scenarios with a high proportion of malicious nodes.

**Keywords:** network function virtualization (NFV); federated learning (FL); blockchain; network traffic prediction; virtual network function migration (VNF migration); dynamic role switching

## 1. Introduction

As one of the core technologies for 5G and future networks, Network Function Virtualization (NFV) greatly improves the flexible management of network resources by turning traditional hardware-based functions into software that can be deployed more easily. NFV supports on-demand Service Function Chaining (SFC), which allows for fast setup and flexible scaling of network functions, paving the way for smarter network management and better service performance [1].

However, in environments with multiple service function providers, the dynamic nature of NFV and the complexity caused by different providers working together make it harder to share data and build joint models. Since these providers usually keep their historical traffic data private and share network resources from the same infrastructure providers, they are often unwilling to share raw data due to privacy concerns, business competition, and regulatory rules [2]. This makes it difficult to combine information from different providers using traditional centralized machine learning, which in turn reduces the accuracy and generalization ability of global traffic prediction models. This problem is especially serious in NFV environments: frequent changes in service function chains, the joining and leaving of providers, and the real-time need for resource management all create a conflict between protecting privacy and keeping computation efficient under centralized approaches. Federated Learning (FL), a distributed learning method, offers a promising solution to privacy issues, allowing participants can train a shared global model without exchanging raw data [3]. Yet existing FL frameworks still struggle with three critical issues: (1) Lack of trust and over-reliance on central

 doi:10.20944/preprints202504.2606.v1



servers: Traditional FL depends on a central server to combine model updates, which becomes a risk if that server is attacked or compromised. In NFV systems with multiple providers, the changing network structure and lack of trust make centralized models hard to apply [4]; (2) No way to track model updates: Traditional FL does not keep a record of the versions between the global and local models. This makes it hard to verify the correctness and development of the models. Malicious nodes can slowly corrupt the global model by uploading slightly altered local models over time. Since most frameworks cannot trace updates back to specific rounds, it's hard to detect and isolate these harmful nodes, which threatens long-term system stability [5]; (3) Hard to adapt to changing environments: In real networks, devices may join or leave at any time, making it difficult for traditional FL systems to adjust model updates and resource use in real time. Delays in communication can cause the model to fail to converge, and the usual update strategies are not flexible enough to handle rapidly changing network traffic, which reduces the model's performance. To address these challenges, we propose a blockchain based verification method that records all model updates and voting results on-chain, ensuring data immutability and trustworthy model updates. This approach helps defend against malicious nodes and improves overall system security. It also enhances the accuracy of network traffic prediction and supports better resource management and service quality for network providers.

To address the above challenges, this manuscript focuses on two key issues in the NFV environment: the reluctance of multiple service function providers to share historical traffic data, and the interference of malicious nodes in federated learning. To this end, we propose a Blockchain and Decentralized Federated Learning Framework with Node Incentive Mechanism for network traffic prediction (BDFL_NI_NTP) in NFV environment. BDFL_NI_NTP achieves a balance between privacy protection and computational efficiency through a decentralized trust mechanism, optimized asynchronous communication, and dynamic role-switching strategies, providing an innovative solution for intelligent network management.

The core contributions include:

1) Modeling the VNF Migration Problem: This manuscript develops a mathematical model for the VNF migration problem under dynamic traffic conditions in NFV networks involving multiple service function providers.

2) Decentralized Federated Learning Mechanism: This manuscript proposes a decentralized FL mechanism based on blockchain that detects and excludes malicious nodes from key roles using a consensus mechanism integrated with a dynamic role management strategy.

3) Asynchronous Communication and Dynamic Role Switching: This manuscript designs an asynchronous aggregation strategy and introduce a role-switching mechanism driven by nodes' historical contributions, helping to balance computational load and improve system stability.

4) Node Join-and-Exit Mechanism: This manuscript introduces a dynamic mechanism to manage node join-and-exit events. The system evaluates node performance based on VNF migration outcomes and applies corresponding reward and punishment mechanisms, thereby enhancing system robustness against malicious behavior.

## 2. Related Work

### *2.1. Network Traffic Forecasting*

Network traffic prediction refers to forecasting future trends and distributions of network traffic by analyzing and modeling historical data. In recent years, various methods combining advanced algorithms with optimization techniques have emerged, significantly improving prediction accuracy and overall system performance. For example, Hou et al. [6] proposed an improved differential evolution algorithm in their study "Fuzzy Neural Network Optimization and Network Traffic Forecasting Based on Improved Differential Evolution." This algorithm optimizes the parameters of fuzzy neural networks, significantly enhancing both prediction accuracy and convergence speed.

Recognizing the advantages of recurrent neural networks (RNNs) in time series forecasting, Ramakrishnan et al. [7], in "Network Traffic Prediction Using Recurrent Neural Networks," explored

 doi:10.20944/preprints202504.2606.v1



the application of RNNs in depth and proposed an effective training strategy to adapt to dynamic changes in complex network traffic. In the field of deep learning for mobile traffic prediction, Huang et al. [8], in "A Study of Deep Learning Networks on Mobile Traffic Forecasting," evaluated the performance of various deep learning architectures and proposed an optimization scheme based on deep learning to address the heterogeneous characteristics of mobile communication data. Considering the potential of graph neural networks (GNNs) in spatiotemporal modeling, Pan et al. [9], in "DC-STGCN: Dual-Channel Based Graph Convolutional Networks for Network Traffic Forecasting," proposed a dual-channel GNN model. This approach captures both spatial correlations and temporal dependencies of network traffic, significantly improving prediction accuracy and model robustness. In addition, Huo et al. [10], in "A Blockchain-Based Security Traffic Measurement Approach to Software Defined Networking," introduced a blockchain-based method to enhance the security and integrity of traffic data in Software Defined Networks (SDN). This approach ensures data immutability and transparency through blockchain technology, enabling decentralized traffic monitoring and management, and effectively preventing malicious tampering of network traffic data.

Qi et al. [11], in "Privacy-Preserving Blockchain-Based Federated Learning for Traffic Flow Prediction," proposed a traffic flow prediction method that integrates privacy preservation, blockchain, and federated learning. The method aims to enhance both prediction accuracy and data privacy. By leveraging blockchain technology, it ensures the transparency and immutability of model parameter sharing among participating nodes during the federated learning process. At the same time, federated learning enables nodes to collaboratively train models without sharing raw data, thus protecting user privacy. Guo et al. [12], in "B2SFL: A Bi-Level Blockchain Architecture for Secure Federated Learning-Based Traffic Prediction," introduced a two-layer blockchain architecture (B2SFL) for secure federated traffic prediction. This architecture ensures data privacy and model security through blockchain mechanisms at both levels. At the lower level, participating nodes train local traffic prediction models using federated learning and record model updates on the blockchain to maintain data confidentiality and model integrity. At the upper level, the blockchain provides a decentralized trust framework to verify node reliability and defend against malicious interference in the prediction process. Kurri et al. [13], in "Cellular Traffic Prediction on Blockchain-Based Mobile Networks Using LSTM Model in 4G LTE Network," proposed a blockchain-based traffic prediction approach for cellular networks that combines blockchain with a Long Short-Term Memory (LSTM) model. This method enhances both the accuracy and security of traffic forecasting in 4G LTE networks. Blockchain technology enables secure storage and sharing of traffic data while ensuring privacy and resistance to tampering. During prediction, LSTM models are employed to capture long-term dependencies in time series data, enabling effective forecasting of future traffic patterns in cellular networks.

### *2.2. Blockchain and Federated Learning*

In recent years, the integration of blockchain and federated learning has garnered increasing attention due to its potential in preserving data privacy, enhancing system trustworthiness, and optimizing resource utilization. Although various approaches have been proposed to combine these technologies, several challenges remain unaddressed.

For instance, Feng et al. [14], in “BAFL: A Blockchain-Based Asynchronous Federated Learning Framework,” introduced a blockchain-enabled asynchronous federated learning framework. The asynchronous update mechanism and the decentralized nature of blockchain alleviates the impact of computation heterogeneity among participants and mitigate training delays. Li et al. [15], in “Blockchain Assisted Decentralized Federated Learning (BLADE-FL): Performance Analysis and Resource Allocation,” addressed the performance bottlenecks in decentralized federated learning and proposed an efficient resource allocation strategy using blockchain. In the field of intelligent healthcare, Połap et al. [16], in “Agent Architecture of an Intelligent Medical System Based on Federated Learning and Blockchain Technology,” developed a multi-agent architecture integrating federated learning and blockchain for collaborative medical data analysis and privacy protection. To

 doi:10.20944/preprints202504.2606.v1



address the communication overhead of traditional federated learning, Cui et al. [17], in "A Fast Blockchain-Based Federated Learning Framework with Compressed Communications," proposed a communication compression mechanism that significantly reduces communication costs and improves model synchronization efficiency. Ren et al. [18], in "BPFL: Blockchain-Based Privacy-Preserving Federated Learning Against Poisoning Attack," proposed a framework that protects local and aggregated model privacy using Paillier encryption with threshold decryption. It also includes incentive mechanisms to penalize malicious nodes and employs blockchain to achieve decentralization. Cosine similarity is used for detecting poisoning attacks, enhancing model robustness on datasets like CIFAR-10. Kasyap et al. [19], in "An Efficient Blockchain-Assisted Reputation-Aware Decentralized Federated Learning Framework," proposed a PoIS (Proof of Interpretation and Selection) consensus mechanism. By using model interpretation and the Shapley value to evaluate client contributions, the framework dynamically selects high-quality nodes and reduces the attack success rate to under 5%.

In contrast, the proposed BDFL_NI_NTP framework in this manuscript introduces several key improvements. First, it eliminates centralized verification nodes by leveraging blockchain consensus mechanisms and smart contracts, thereby removing single points of failure. Second, it introduces a contribution-based role-switching strategy and a dynamic node join/exit mechanism to enable efficient resource allocation and isolate malicious participants. Finally, by combining blockchain based record keeping with a majority-voting scheme, it ensures the immutability of model parameters. A reward and punishment mechanism based on contribution evaluation further incentives high-quality node participation and suppresses malicious behavior.

## 3. System Architecture

### *3.1. Mathematical Model of VNF Migration Problem*

#### 3.1.1. Decision Variable

A binary decision variable $x_{s,v}$ is used to indicate whether to migrate VNF $f_{s,v}$ or not.

$$x_{s,v} = \begin{cases} 1 \ \ if \ \ VNF \ \ f_{s,v} \in F_s \ \ is \ \ migrated, \\ 0 \ \ otherwise. \end{cases} \tag{1}$$

A binary decision variable $y_{s,v,n}$ is used to indicate whether to map VNF $f_{s,v}$ to a physical node $n$.

$$y_{s,v,n} = \begin{cases} 1 \ \ if \ \ VNF \ \ f_{s,v} \ \ is \ \ embedded \ \ on \ \ node \ \ v_n, \\ 0 \ \ otherwise. \end{cases} \tag{2}$$

A binary decision variable $z_{s,v,ll'}$ is used to indicate whether a virtual link $l_{s,\theta_{s,v}}$ is mapped to a physical link $e_{l,l'}$.

$$z_{s,v,ll'} = \begin{cases} 1 \ \ if \ \ VL \ \ l_{s,\theta_{s,v}} \ \ is \ \ mapped \ \ to \ \ PL \ \ e_{l,l'}, \\ 0 \ \ otherwise. \end{cases} \tag{3}$$

#### 3.1.2. Constraints

The optimization problem is constrained by conditions (1) to (13).

Constraint (1):

It is guaranteed that each VNF for an accepted SFC request will be processed only once on an NFV node and will only run on an NFV node that supports that VNF.

 doi:10.20944/preprints202504.2606.v1



$$\sum_{v_n \in V^{ser}} y_{s,v,n} \cdot b_{n,v} = 1, \forall r_s \in R, f_v \in F_s. \tag{4}$$

Where $b_{n,v}$ is a binary decision variable that takes the value of 1 if VNF $f_{s,v}$ can be instantiated on physical node $v_n$; otherwise, it takes the value of 0.

Constraints (2) and (3):

They are used to guarantee that each accepted SFC request starts at the start node and ends at the target node.

$$y_{s,\theta'_{s,0},n_s^{src}} = 1, \forall r_s \in R. \tag{5}$$

$$y_{s,\theta'_{s,|F_s|+1},n_s^{dst}} = 1, \forall r_s \in R. \tag{6}$$

Where $\theta_{s,v}$ denotes the location sequence of VNF $f_{s,v}$, $\theta'_{s,q}$ denotes the $q$ th VNF of SFC request $r_s$, $\theta'_{s,0}$ denotes the start node of SFC request $r_s$, $\theta'_{s,|F_s|+1}$ denotes the end node of SFC request $r_s$, $n_s^{src}$ denotes the start physical node of SFC request $r_s$, and $n_s^{dst}$ denotes the end physical node of SFC request $r_s$.

Constraint (4):

Ensure that the traffic of each SFC request cannot be split.

$$\sum_{v_m \in N_n} z_{s,v,nm} + \sum_{v_m \in N_n} z_{s,v,mn} \le 1, \forall r_s \in R, f_{s,v} \in F_s, v_n \in V^{ser}, e_{m,n} \in E, m < n. \tag{7}$$

Where $z_{s,v,mn}$ is a binary decision variable that takes the value of 1 if virtual link $l_{s,\theta_{s,v}}$ is deployed on physical link $e_{m,n}$ and 0 otherwise.

Constraint (5):

Ensure that each accepted SFC request has a conserved traffic flow at each node.

$$\sum_{v_m \in N_n} z_{s,v,nm} - \sum_{v_m \in N_n} z_{s,v,mn} = y_{s,v,n} - y_{s,\theta'_{s,(\theta_{s,v}-1)},n}, \quad \forall r_s \in R, f_{s,v} \in F_s, v_n \in V. \tag{8}$$

Where $\theta_{s,v}$ denotes the sequence of locations of VNF $f_{s,v}$ and $\theta'_{s,q}$ denotes the $q$ th VNF requested by the SFC. $\theta'_{s,(\theta_{s,v}-1)}$ then represents the previous VNF for which the SFC requests $r_s$.

Constraint (6):

Guarantee not to exceed the processing capacity of each VNF.

$$\sum_{r_s \in R} b_{s,v} \cdot b_{n,v} \cdot y_{sv,ni} \cdot c_{s,v}^{pro} \le c_{v,i}^{pro}, \forall v_n \in V^{ser}, f_v \in F, f_{v,i} \in I_{n,v}. \tag{9}$$

Where $b_{s,v}$ is a binary decision variable that takes the value of 1 if VNF $f_v$ belongs to SFC request $r_s$; otherwise, it takes the value of 0. $y_{sv,ni}$ is also a binary decision variable that takes the value of 1 if VNF $f_{s,v}$ is mapped to an instance $f_{v,i}$ of a physical node $v_n$; otherwise, it takes the value of 0. $c_{s,v}^{pro}$ denotes the processing demand of VNF $f_{s,v}$, and $c_{v,i}^{pro}$ denotes the processing capacity of the instance $f_{v,i}$.

Constraint (7):

 doi:10.20944/preprints202504.2606.v1



Ensure that the delay constraints for each SFC flow are not violated.

$$\sum_{\theta_{s,v}=0}^{|F_s|}\sum_{e_{m,n}\in E}\left(\frac{l_s}{c_s^{bw}}+d_{m,n}^{paga}\right)\cdot z_{s,v,mn}+\sum_{f_{s,v}\in F_s}\sum_{v_n\in V^{ser}}\left(\frac{1}{c_{s,v}^{pro}}+\frac{1}{c_{s,v}^{pro}-\lambda_s}\right)\cdot y_{s,v,n}\le d_s^{\max},\ \forall r_s\in R. \tag{10}$$

Where $l_s$ denotes the packet size of the SFC request $r_s$, $c_s^{bw}$ denotes the bandwidth requirement of the SFC request $r_s$, each physical link $e_{m,n}\in E$ has a link propagation delay $d_{m,n}^{paga}$, $c_{s,v}^{pro}$ denotes the processing rate of the VNF $f_{s,v}$, $\lambda_s$ denotes the traffic arrival rate of the SFC request $r_s$, and $d_s^{\max}$ denotes the end-to-end delay requirement of the SFC request $r_s$.

Constraint (8):

Ensure that the bandwidth capacity of each link is not exceeded.

$$\sum_{r_s\in R}\sum_{f_v\in F_s}\left(z_{s,v,ll'}+z_{s,v,l'l}\right)\cdot c_s^{bw}\le c_{l,l'}^{bw},\forall e_{l,l'}\in E,l<l'. \tag{11}$$

Where each link $e_{l,l'}\in E$ possesses bandwidth capacity $c_{l,l'}^{bw}$.

Constraint (9):

Ensure that the processing capacity of each NFV node is not exceeded.

$$\sum_{f_v\in F} b_{n,v}\cdot n_{n,v}\cdot c_{v,i}^{pro}\le c_n^{pro},\forall v_n\in V^{ser},n_{n,v}\le n_{n,v}^{\max}. \tag{12}$$

Where $n_{n,v}$ denotes the number of instances of VNF $f_v$ in NFV node $v_n$, $c_n^{pro}$ denotes the processing power of NFV node $v_n\in V^{ser}$, and $n_{n,v}^{\max}$ denotes the maximum number of VNF $f_v$ instances in NFV node $v_n$.

Constraint (10):

It is guaranteed that the computational capacity of each NFV node is not exceeded.

$$c_n^{compused}=\sum_{r_s\in R}\sum_{f_v\in F_s} y_{s,v,n}\cdot c_{s,v}^{comp}+\sum_{f_v\in F} b_{n,v}\cdot n_{n,v}\cdot c_{v,i}^{comp}\le c_n^{comp},\ \forall v_n\in V^{ser},n_{n,v}\le n_{n,v}^{\max}. \tag{13}$$

Where $c_{s,v}^{comp}$ denotes the basic CPU resource consumption of VNF $f_{s,v}$ at the physical node, $c_{v,i}^{comp}$ denotes the amount of CPU resources required by VNF instance $f_{v,i}$, and $c_n^{comp}$ denotes the CPU resource capacity of node $v_n\in V^{ser}$.

Constraint (11):

The storage capacity of each NFV node is guaranteed not to be exceeded.

$$\sum_{r_s\in R}\sum_{f_v\in F_s} y_{s,v,n}\cdot c_{s,v}^{stor}+\sum_{f_v\in F} b_{n,v}\cdot n_{n,v}\cdot c_{v,i}^{stor}\le c_n^{stor},\forall v_n\in V^{ser},n_{n,v}\le n_{n,v}^{\max}. \tag{14}$$

 doi:10.20944/preprints202504.2606.v1



Where $c_{s,v}^{stor}$ denotes the base demand for storage resources by VNF $f_{s,v}$ on an NFV node, $c_{v,i}^{stor}$ denotes the amount of storage resources required by VNF instance $f_{v,i}$, and $c_n^{stor}$ denotes the total storage resource capacity of node $v_n \in V^{ser}$.

3.1.3. Migration Cost

(1) Delay Cost

Migration Delay: The shorter the delay of VNF migration, the greater the return obtained. Therefore, we calculate the migration cost based on the migration delay. Each VNF to be migrated transmits the VNF operational state and other information stored in the original physical node to the new physical node. The migration latency of VNF includes propagation latency and transmission latency. These two types of delays are analyzed below.

Propagation delay is the delay that occurs when packets propagate in the medium and depends mainly on the propagation distance and propagation speed. $c^{paga}$ denotes the propagation speed and $d_{l,l'}$ denotes the distance of the physical link $e_{l,l'}$. Thus, the propagation delay between NFV node $v_n$ to NFV node $v_{n'}$ is defined as follows:

$$d_{sv,nn'}^{paga} = \sum_{e_{l,l'} \in E} z_{s,v,ll'} \cdot d_{l,l'}^{paga} = \sum_{e_{l,l'} \in E} z_{s,v,ll'} \cdot \frac{d_{l,l'}}{c^{paga}}. \tag{15}$$

The transmission delay is the delay incurred when transmitting a packet and is related to the packet size and the transmission bandwidth. $c^{migbw}$ to represent the transmission bandwidth for migrating VNFs between two NFV nodes. The transmission delay $d_{sv,nn'}^{tran}$ from NFV node $v_n$ to NFV node $v_{n'}$ is defined as follows:

$$d_{sv,nn'}^{tran} = \frac{c_{s,v}^{stor}}{c^{migbw}} \cdot \sum_{e_{l,l'} \in E} z_{s,v,ll'}. \tag{16}$$

The migration delay for each VNF can be expressed as follows.

$$d_{s,v}^{mig} = d_{sv,nn'}^{tran} + d_{sv,nn'}^{paga}. \tag{17}$$

The migration cost at time slice alternation is defined as follows.

$$c^{mig} = \omega \cdot \sum_{r_s \in R} \sum_{f_v \in F_s} x_{s,v} \cdot d_{s,v}^{mig}. \tag{18}$$

Where $\omega$ is the unit cost of migration delay.

End-to-end delay: The shorter the end-to-end delay of the SFC, the greater the reward obtained. The deployment cost is calculated based on the end-to-end delay of the SFC.

The end-to-end delay of SFC includes transmission delay, propagation delay, processing delay, and queuing delay.

$$d_s = \sum_{\theta_{s,v}=0}^{|F_s|} \sum_{e_{m,n} \in E} \left( \frac{l_{s,v}}{c_s^{bw}} + d_{m,n}^{paga} \right) \cdot z_{s,v,mn} + \sum_{f_{s,v} \in F_s} \sum_{v_n \in V^{ser}} \left( \frac{1}{c_{s,v}^{pro}} + \frac{1}{c_{s,v}^{pro} - \lambda_s} \right) \cdot y_{s,v,n}. \tag{19}$$

The deployment cost for all SFC requests is defined as follows.

 doi:10.20944/preprints202504.2606.v1



$$c^{place} = \mu \cdot \sum_{r_s \in R} x_s \cdot d_s. \quad (20)$$

Where $\mu$ is the unit cost of deployment delay.

Delay cost: The delay cost at time slice alternation is the sum of end-to-end delay and migration delay:

$$c^{migrate} = c^{mig} + c^{place}. \quad (21)$$

(2) Operating Cost

Energy consumption is an important factor that should not be ignored in the network operation cost. Compared with node energy consumption, link energy consumption is relatively small, so we only consider the energy consumption of NFV nodes. In addition, the energy consumption of nodes is mainly related to CPU utilization. Therefore, the energy consumption of a physical node is defined as follows.

$$e_n = \varepsilon_0 + (\varepsilon_1 - \varepsilon_0) \cdot \frac{c_n^{compused}}{c_n^{comp}}. \quad (22)$$

Where $\varepsilon_0$ is the basic energy consumption of the NFV node and $\varepsilon_1$ is the maximum energy consumption of the NFV node.

The running cost between two migrations is defined as follows.

$$c^{operate} = \gamma \cdot l_t^{mig} \cdot \sum_{v_n \in V^{ser}} e_n. \quad (23)$$

Where $\gamma$ is the unit energy cost in network operation, and $l_t^{mig}$ is the length of time between two migrations.

(3) Rejection Cost

The rejection cost is calculated based on the resource demand of each rejected SFC request, which is calculated as follows:

$$c^{punish} = \mu_1 \cdot \sum_{r_s \in R} \sum_{f_v \in F_s} c_{s,v}^{comp} + \mu_2 \cdot \sum_{r_s \in R} \sum_{f_v \in F_s} c_{s,v}^{stor} + \mu_3 \cdot c_s^{bw} \cdot (|F_s| + 1). \quad (24)$$

Where $\mu_1$, $\mu_2$ and $\mu_3$ are the unit penalty costs.

3.1.4. Optimization Objectives

The SFC migration (SFCM) problem is formulated as the following integer linear programming (ILP) with the objective of minimizing cost:

$$\begin{aligned} &Minimize: c^{mig} = c^{operate} + c^{migrate} + c^{punish}, \\ &Subject\ \ to: Constrains\ \ (1)-(11). \end{aligned} \quad (25)$$

Since the problem is NP-hard, it is difficult to solve using an exhaustive search algorithm.

*3.2. System Model*

3.2.1. General Architecture

The set of network service function providers participating in federated learning is denoted as $D = \{d_1, d_2, ..., d_f\}$. Based on the node's historical contribution value, all nodes are categorized into

 doi:10.20944/preprints202504.2606.v1



four different roles[20], which are (1) Worker nodes $W=\{w_1, w_2, ..., w_n\}$: perform the training task of the local model using its own private historical network traffic, and upload the end-of-training model (shared in the form of model parameter); (2) Residual nodes $R=\{r_1, r_2, ..., r_p\}$: perform the training task of the local model using its own private historical network traffic, and upload the updated model. They do not participate in the final model aggregation, but Evaluator and Aggregator will calculate the contribution value; (3) Evaluator nodes $E=\{e_1, e_2, ..., e_m\}$: evaluate the models uploaded by the Worker nodes as well as the residual nodes, and submit the evaluation results to Aggregator; (4) Aggregator node $A=\{a_1\}$: Collects the qualified models, completes the global model aggregation, and updates, generates and uploads blocks. For the number of Worker nodes and Evaluator nodes, the condition needs to be satisfied $n>m>1$; for the number of the four types of nodes, the condition needs to be satisfied $n+m+p+1=f$ .

Among them, one block consists of the following parts: (1) Block header: provides information on the chain structure and generation time of the blockchain, ensuring data integrity and traceability; (2) Global model information: records the aggregated model parameters of federated learning, ensuring the consistency and transparency of the global model; (3) Node participation information: stores node's contribution value to form a positive incentive mechanism to attract high-quality node participation ; (4) Transaction records: track the model parameters uploaded by nodes and their verification results to ensure the authenticity of the data; (5) Committee results: provide the basis for evaluating the model parameters to enhance the fairness and security of the system; (6) Block signatures: ensure the security of the block content and prevent tampering.

3.2.2. Implementation Steps

(1) Rejection Cost

The implementation steps of the BDFL_NI_NTP scheme is shown in Figure 1.

 doi:10.20944/preprints202504.2606.v1



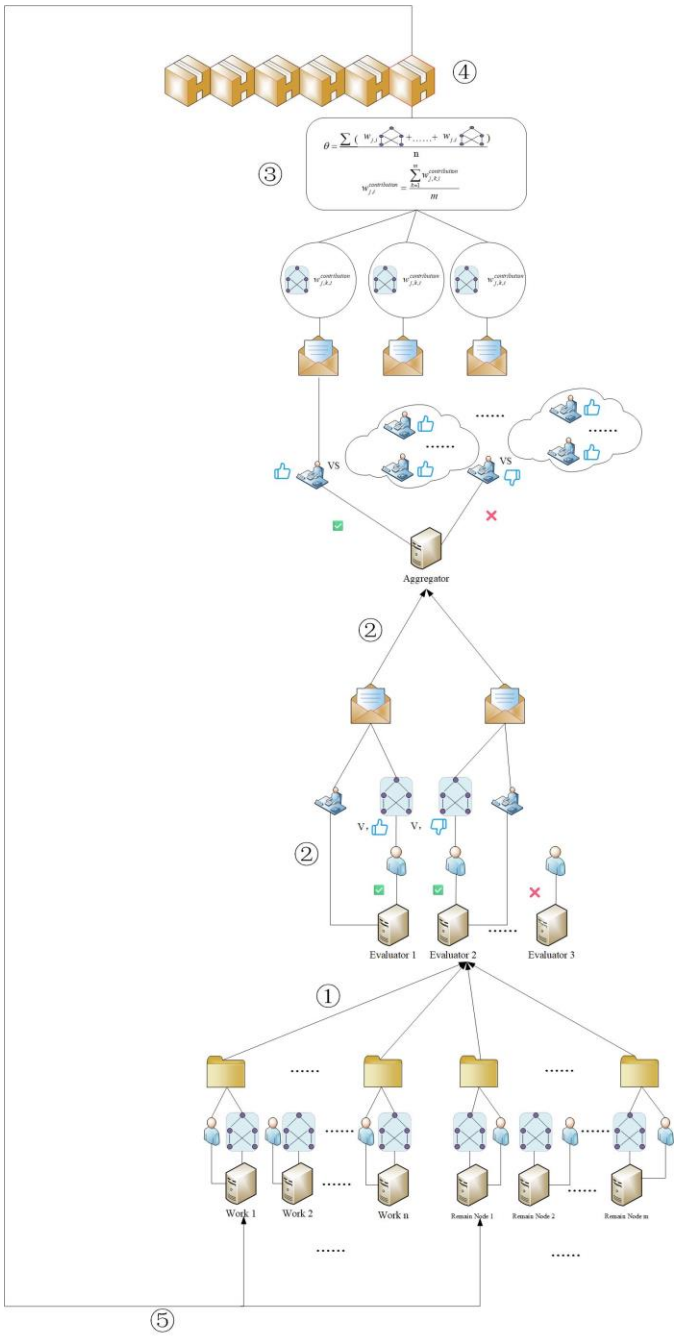


**Figure 1.** Blockchain and Decentralized Federated Learning.

①In the $j$ th training round of federated learning, Worker node $w_i$ and Residual node $r_i$ train the local model using their private historical network traffic data, and the parameters of the model generated by the training are denoted as $L_{j,i}^{para}$. Then, a transaction $T_{j,i}^{transact}$ containing the model parameters $L_{j,i}^{para}$ and the identity certificate is generated, and the transaction $T_{j,i}^{transact}$ is sent to all Evaluator nodes for evaluation.

②Evaluator node $e_k$ receives the transaction, verifies the identities of $w_i$ or $r_i$, and extracts the model parameters $L_{j,i}^{para}$ for the nodes for evaluation. Evaluator node $e_k$ calculates the migration cost according to Eq.(25) for the model parameters uploaded by nodes $w_i$ or $r_i$, using local historical network traffic data. Compared with the best model with minimum migration cost, if the migration cost of the model does not exceed the threshold, the node is judged to be qualified, and it is rewarded and updated with the contribution value of the model; if not qualified, it is penalized

 doi:10.20944/preprints202504.2606.v1



and updated with the contribution value of the model. Then, pack the validation result of the model parameter $L_{j,i}^{para}$, the contribution value of the model, and the ID certificate of the Evaluator node $e_k$ to form a transaction $T_{j,k}^{transact}$, and send the transaction $T_{j,k}^{transact}$ to the Aggregator node $a_1$.

③Aggregator node $a_1$ receives the transactions from all Evaluator nodes, and if an Evaluator and the majority of Evaluators have the same evaluation results, the result of its evaluation of the contribution value is taken into account; otherwise, the judgment of the Evaluator is ignored. If the majority of evaluators assess the model which has parameters $L_{j,i}^{para}$ as passed, the model which has parameters $L_{j,i}^{para}$ will be included in the aggregation process. Meanwhile, for all the Worker nodes and Residual nodes, according to the contribution value judged by Evaluators, the average value is taken as the final value of contribution for each Worker and Residual node.

④ The Aggregator node performs model aggregation using Worker nodes which are evaluated as passed in the aggregation process to generate the global model with parameters $G_j^{para}$ as well as the block $B_{j,a}^{block}$.

⑤ Propagate the global model to all Workers and Residual nodes to ensure the next round of federated learning.

In the whole process described above, the evaluation result of Evaluator is supervised by Aggregator, and the aggregation process of Aggregator is supervised by the consensus mechanism.

(2) Aggregation Method

The aggregation of Aggregator is to update the global model as follows:

$$G_j^{para} = \frac{\sum_{i=1}^{n_j^{aggregate}} w_{j,i} \cdot L_{j,i}^{para}}{n_j^{aggregate}}. \quad (26)$$

Where $n_j^{aggregate}$ is the number of Worker nodes participating in the aggregation; $w_{j,i}$ is the weight factor of the local model $L_{j,i}^{para}$, and the model with a larger predicted traffic value will be given a larger weight. Thus, the weight factor is calculated as:

$$w_{j,i} = \frac{v_i^{predict}}{\frac{n_j^{aggregate}}{\sum_{q=1} v_q^{predict}}}. \quad (27)$$

Where $v_i^{predict}$ denotes the predicted traffic value of Worker node $w_i$.

(3) Role switching mechanism

In large-scale distributed networks, unfair switching of nodes may cause the model to deviate from normal updating during federated learning, in addition, frequent node switching can lead to resource waste and performance degradation [21]. It is important to be able to ensure the stability and efficiency of the system, but also to be able to quickly identify and punish nodes with malicious behaviors and reward nodes that work normally. Therefore, it is especially important to develop a reasonable role switching mechanism for nodes.

Initialization: First of all, in the initial state, there is no way to examine the working ability and working attitude of each node. Therefore, roles can be randomly assigned to nodes. After the initial role assignment is completed, the federated learning will be repeated.

Evaluation of Worker and Residual nodes: Then, the Worker node and the Residual nodes complete the training task locally and upload the trained model to the Evaluator node. The Evaluator node $e_k$ evaluates the model parameters uploaded by the nodes using the local historical network

 doi:10.20944/preprints202504.2606.v1



traffic data, and uses Eq. (25) to calculate the migration cost $c_{k,i}^{mig}$ using the predicted value of the model. The gap ratio is obtained by comparing the Worker node $w_i$ or the Residual node $r_i$ with the minimum value of migration cost as shown in the following equation:

$$c_{k,i}^{ratio} = \frac{c_{k,i}^{mig} - c_k^{\min mig}}{c_k^{\min mig}} \quad (28)$$

Where, $c_k^{\min mig}$ is the minimum value of migration cost calculated by Evaluator node $e_k$ for the current Worker node and the set of Residual nodes.

When the gap ratio $c_{k,i}^{ratio}$ is greater than the threshold $\theta_1$, the current node is considered to have some malicious behavior. So, it is marked as a malicious behavior node and its contribution value is reduced as shown in the following equation:

$$w_{j,k,i}^{contribution} = w_{j-1,k,i}^{contribution} - \alpha_1 \quad (29)$$

Where, $\alpha_1$ is a constant value.

When the disparity ratio $c_{k,i}^{ratio}$ is less than the threshold value $\theta_1$, the current node is considered to have no malicious behavior. Therefore, it is marked as a normal behavior node and its contribution value is increased as shown in the following equation:

$$w_{j,k,i}^{contribution} = w_{j-1,k,i}^{contribution} + \alpha_2 \quad (30)$$

Where, $\alpha_2$ is constant value.

Evaluation of Evaluator nodes: Evaluator will give the evaluation results of Worker nodes and Residual nodes to Aggregator, which is used to evaluate Evaluator for malicious behavior. If the number of Evaluator with malicious behavior is high, the role switching process will be triggered. In addition, in order to avoid the system instability caused by frequent role switching, we set that the role replacement will also be performed once every $T$ mini-round. During role switching, both normal working Aggregator nodes and Evaluator nodes no longer take up their original roles, they are reassigned contribution values and roles instead, if an Aggregator or Evaluator node has malicious behavior, its contribution value will be set to a smaller value, and its role will be switched to a Residual node. If the judgment of Worker node is more reasonable, it is considered that it does not have malicious behavior. The degree of reasonableness of Evaluator judgment can be obtained based on the comparison of the judgment results with other Evaluators. The evaluation is calculated as follows for Evaluator $e_k$:

$$S_k = \frac{\sum_{i=1}^{n}\sum_{j\neq k}\mathbb{Z}\left(V_{j,i},V_{k,i}\right)+\sum_{i=1}^{p}\sum_{j\neq k}\mathbb{Z}\left(V_{j,i},V_{k,i}\right)}{n\cdot\left(m-1\right)+p\cdot\left(m-1\right)} \quad (31)$$

Where $V_{j,i}$ is the result of Evaluator $e_j$'s vote for Worker node $w_i$ or Residual node $r_i$. When the vote is qualified, the value is 1, otherwise, the value is 0. $\mathbb{Z}\left(x,y\right)$ is the similarity function, when the values of $x$ and $y$ are the same, the function value is 1, otherwise, the function value is 0.

When the value of $S_k$ is greater than the threshold $\theta_2$, the evaluation result of the Evaluator is considered valid, and at the same time, it is considered that the Evaluator has no malicious

 doi:10.20944/preprints202504.2606.v1



behavior; otherwise, it is considered that its evaluation result is invalid, the Evaluator has malicious behavior, and it is necessary to mark it as a Residual node. When it starts to train the model in the next round, it can't participate in the model aggregation. When the number of Evaluator nodes demoted as Residual nodes exceeds half of the maximum number of Evaluators, this round of federated learning is terminated and the role switching process is started.

The evaluation results (e.g., model parameter scores, pass marks, etc.) of all Evaluators on Workers and Residual nodes are publicly recorded via the blockchain, and the recorded data are transparent and tamper-proof. Meanwhile, their evaluation results are also sent to Aggregator, which performs cross-validation based on the transactions uploaded by each Evaluator, and adopts Eq. (31) to calculate the similarity. Thereby, enabling the Aggregator to supervise and manage the Evaluators based on their evaluation results.

Actual contribution values of Worker and Residual nodes: Finally, Aggregator calculates the actual contribution value of all Worker nodes and Residual nodes, i.e., the contribution value evaluated by all Evaluator nodes with normal behavior is averaged as shown in the following equation:

$$w_{j,i}^{contribution} = \frac{\sum_{k=1}^{m} w_{j,k,i}^{contribution}}{m} \quad (32)$$

Where, $m$ is the current number of Evaluator nodes.

*3.3. Consensus Mechanism*

Aggregator nodes may also have the possibility of intentionally tampering with the aggregation process, such as intentionally excluding certain valid models, or inserting malicious models. At this point, a consensus mechanism is needed to verify and supervise the aggregation process of Aggregator. For example, The malicious behavior of an Aggregator node is determined by verifying if the list of qualified nodes selected by the Aggregator matches the results from non-malicious Evaluator nodes. Specifically, the model evaluation results (e.g., the list of qualified nodes) of all non-malicious Evaluator nodes on Worker nodes are publicly stored in the blockchain to ensure that the data cannot be tampered with. The Aggregator node is required to synchronize the submission of the list of qualified Worker nodes selected for this round of aggregation when generating a block. The consensus mechanism compares the list of qualified nodes submitted by both parties, and if it is consistent, the Aggregator is considered to have no malicious behavior; if it is inconsistent, the Aggregator is considered to have malicious behavior and triggers the process of role switching.

In summary, the consensus mechanism consists of the following steps: (1) Worker nodes upload local model parameters with their own identifiers to ensure the credibility of the data. (2) The Evaluator node independently verifies the model and decides whether to accept the model through majority voting (e.g., more than half support). The voting result is recorded to the blockchain through a smart contract to ensure that it cannot be tampered with. (3) The Aggregator node collects the verified parameters and asynchronously aggregates them to generate the global model.

Through the above mechanism, the system achieves efficient and secure consensus in a decentralized environment, while balancing the conflict between privacy protection and model performance.

## 4. Algorithms

This section describes the proposed algorithms in detail.

**Algorithm 1 :Blockchain and Decentralized Federated Learning Fra**mework with Node Incentive. .

 doi:10.20944/preprints202504.2606.v1



**Inputs:** Node set $N$ , Block $B_{j,a}^{Block}$ and Global model $G_j^{para}$ .

**Output:** New block $B_{j+1,a}^{Block}$ , New global model $G_{j+1}^{para}$ .

For each node i:
    Randomize the type to which the nodes belongs.
End for
For each node i:
    If node i belongs worker:
        $w_{j,i}^{contribution} \leftarrow 0$;
    End if
End for
For each round out_r from 1 to EPOCH_OUT:
    For each round in_r from 1 to EPOCH_IN:
        Train each local model using local private historical traffic data;
        //At Worker
        Generate transactions with model parameter of each local model and authentication information, then send them to all Evaluator nodes; // At Worker
        For each model i: //At Evaluator
            If model i is worker or remain node:
                For each Evaluator $e_k$ :
                    Verify the identity of the node;
                    If identity passes:
                        Evaluate the migration cost of the model i according to Eq.(25);
                        Compute the gap ratio $c_{k,i}^{ratio}$ according to Eq.(28);
                        If $c_{k,i}^{ratio}$ < threshold $\theta_1$ :
                            Select the node and update the contribution according to Eq.(30);
                        Else:
                            Ignore the node and update the contribution according to Eq.(29);
                        End if
                    End if
                End for
            End if
        End for
        For each node k://At Aggregator
            If node k is evaluator:
                Compute evaluation $S_k$ for evaluator $e_k$ according to Eq.(31);

 doi:10.20944/preprints202504.2606.v1



| | |
|---|---|
| 32 | If evaluation $S_k$ > threshold $\theta_2$: |
| 33 | Generate a transaction $T_{j,k}^{transact}$ containing the validation results and send it to the Aggregator node. |
| 34 | Else: |
| 35 | Set evaluator to be the remaining node; |
| 36 | Set the contribution value of the remaining node k to 0; |
| 37 | If the number of evaluator nodes is less than half the maximum number of evaluator nodes. |
| 38 | TRANS=1; |
| 39 | Else: |
| 40 | TRANS=0; |
| 41 | End if |
| 42 | End if |
| 43 | End if |
| 44 | End for |
| 45 | For each Worker:// At Aggregator |
| 46 | If more than half of the Evaluators select this worker. |
| 47 | Select this worker as the aggregation node; |
| 48 | End if |
| 49 | End for |
| 50 | For each worker:// At Aggregator |
| 51 | Calculate the actual contribution according to Eq.(32); |
| 52 | End for |
| 53 | Generate global model $G_{j+1}^{para}$ and new block $B_{j+1,a}^{Block}$ according to Eq.(26) using all of local models of the selected aggregator; |
| 54 | Broadcast the new block $B_{j+1,a}^{Block}$ to all Workers and the Remaining nodes. |
| 55 | If TRANS==1: |
| 56 | break; |
| 57 | End if |
| 58 | End for |
| 59 | Reassign node roles according to **Algorithm 2**; |
| 60 | End for |

**Algorithm 2: Node Rol**e Assignment and Switching.

| |
|---|
| **Input:** Current roles of all nodes. |
| **Output:** Updated node roles (Aggregator, Worker, Residual, or Evaluator nodes). |
| 1 Select a node as aggregator from the Workers and Residual nodes with the highest contribution; |

 doi:10.20944/preprints202504.2606.v1



| | |
|---|---|
| 2 | Select k nodes as evaluators from the Workers and Residual nodes with higher contribution; |
| 3 | for each node i: |
| 4 | If the node i is the former Evaluator or Aggregator: |
| 5 | set the node i as Worker; |
| 6 | End if |
| 7 | End for |
| 8 | Select n-m-1 nodes as Workers from the Workers and Residual nodes with higher contribution; |
| 9 | Set the remaining nodes as Residual nodes; |
| 10 | Return the result of the updated role assignment; |

## 5. Simulation

### *5.1. Settings*

In this section, we use Python 3.6 and Pytorch to verify the proposed scheme. The simulation platform is built on an Intel Core computer with a 1.8 GHz central processor and 8 GB Random Access Memory.

#### 5.1.1. Parameter Settings

In the simulation, we obtain the network topology with scale: [200,991], where the numbers represent the quantity of NFV enabled nodes, and physical links, respectively. The CPU capacity of nodes ranges from 100 to 200 cores, the storage capacity of nodes ranges from 100 to 200, the processing capacity of nodes ranges from 80 to 150, the link bandwidth capacities between two nodes ranges from 15 to 25 Mbps, and the link latency between two adjacent nodes ranges from 1 to 2s. The experiment considers 8 types of VNFs. Each SFC randomly selects a few VNFs, and the length of each SFC is between 2 and 6. The SFC latency limit ranges from 30 to 50s. There are 10 SFC requests in the network. The processing latency of VNF in service node is from 2 to 4s. CPU requirement of each VNF is from 1 to 2 cores, storage requirement of each VNF is from 1 to 2, processing resource requirement of each VNF is from 1 to 2, and the bandwidth resource requirement of each virtual link is from 10 to 20bps. In addition, the number of network service providers is set to 12.

Each local model contains 2-layer LSTM, with each hidden layer having 50 neurons. The input layer's number of neurons corresponds to the state-space dimension, followed by two hidden layers, then the dueling network structure connected to the output layer. The output layer's number of neurons corresponds to the discrete action space dimension. The algorithm uses SoftPlus as the activation function, with a maximum iteration number of 10.

Set up 12 federated learning participants, including 7 Workers (accounting for 58% of the total nodes), responsible for local model training and update uploads; 4 Evaluators (accounting for 33% of the total nodes), responsible for model parameter validation; and 1 Aggregator (accounting for 9% of the total nodes), responsible for global model aggregation.

Other parameter settings are shown in **Error! Reference source not found.**.

**Table 1.** Parameter settings.

| Hyper parameters | Values |
|---|---|
| Learning rate | 0.0001 |
| Dropout factor | 0.2 |
| The number of outer layer cycles | 10 |
| The number of inner layer cycles | 5 |
| $\omega$ in Eq. (18) | 20 |

 doi:10.20944/preprints202504.2606.v1



| | |
|---|---|
| $\mu$ in Eq. (20) | 1 |
| $\varepsilon_0$ in Eq. (22) | 200 |
| $\varepsilon_1$ in Eq. (22) | 300 |
| $\gamma$ in Eq. (23) | 0.01 |
| $\mu_1$ in Eq. (24) | 1 |
| $\mu_2$ in Eq. (24) | 1 |
| $\mu_3$ in Eq. (24) | 1 |
| $\alpha_1$ in Eq. (29) | 100 |
| $\alpha_2$ in Eq. (30) | 100 |
| $\theta_1$ | 1 |
| $\theta_2$ | 50% |

5.1.2. Data Sources

Network traffic data: Divide the network traffic data into time windows of 60 minutes each, and extract time-series features (e.g., traffic peaks). After standardization, split the data into a training set (60%), validation set (20%), and test set (20%).

NFV simulation data: Simulate a dynamic NFV environment to perform operations such as SFC mapping, VNF migration, and dynamic node joining and exiting.

5.1.3. Benchmark Schemes

We compare the performance of our proposed BDFL_NI_NTP approach against the following two baseline schemes. All three schemes are implemented using identical settings. More specifics regarding the two baseline schemes are as follows:

(1) Baseline 1, FL: This is a network traffic prediction scheme based on a centralized server-based federated learning (FL) framework without a validation mechanism, where all local model parameters are directly aggregated [22].

(2) Baseline 2, BlockFL: BlockFL is a decentralized architecture combining blockchain and federated learning, which adopts a committee-based validation mechanism for intermediate parameters. After training, nodes upload updates and trigger aggregation operations [23]. This model does not incorporate a role-switching mechanism.

*5.2. Evaluation Indicators*

The following performance metrics are used to evaluate the performance of the algorithm:

(1) Loss: It represents the average MSE loss of the predicted network traffic. It is defined as follows.

$$c^{averae,migrate} = \frac{\sum_{i=1}^{n} l_i}{n}. \quad (33)$$

Where $l_i$ is MSE loss between the prediction value and actual value for the i-th data group, and $n$ is the total number of data points.

(2) Latency: It represents the average migration latency of the physical network. It is defined as follows.

$$c^{averae,migrate} = \frac{\sum_{i=1}^{n} c^{migrate}}{n}. \quad (34)$$

 doi:10.20944/preprints202504.2606.v1



Where $c^{migrate}$ is analyzed in Eq.(21).

(3) Energy: The average amount of energy consumption of the network, which consists of energy consumption of all servers of the accepted SFC requests. It is defined as follows.

$$c^{averae,operate} = \frac{\sum_{i=1}^{n} c^{operate}}{n}. \quad (35)$$

Where $c^{operate}$ is defined in Eq.(23).

(4) Punish: It is the resource required by the failed SFC requests. An SFC request is said to be failed when it can not be embedded successfully. It can be defined as follows.

$$c^{sum,punish} = \sum_{i=1}^{n} c^{punish}. \quad (36)$$

Where $c^{punish}$ is defined in Eq.(24).

(5) Cost: It is the linear combination of the above three metrics, and it is defined as following.

$$c^{averae,mig} = \frac{\sum_{i=1}^{n} c^{mig}}{n}. \quad (37)$$

Where $c^{mig}$ is calculated by Eq.(25).

*5.3. Performance Evaluation*

To evaluate the effectiveness of BDFL_NI_NTP, we train the models using the parameters selected in the aforementioned analysis and deploy them in the federated learning (FL) module. The proportion of malicious nodes is uniformly set between 16% and 83%. Initially, we analyze the experimental results with 6 malicious nodes (50% of the total).

During the training process of our proposed BDFL_NI_NTP framework, we record the loss values. After multiple training rounds, the variation in loss values is illustrated in **Error! Reference source not found.**.

To comprehensively evaluate the training performance, we increased the number of iterations with the outer loop set to 14 cycles and the inner loop to 20 cycles. We record the reward values of the validation data during the training process. As can be seen in Figure 1. Decentralized federated learning, our BDFL_NI_NTP algorithm has a rapid convergence trend. In fact, after 50 rounds of training, the reward values obtained during the validation process in the remaining rounds is already very close.

 doi:10.20944/preprints202504.2606.v1



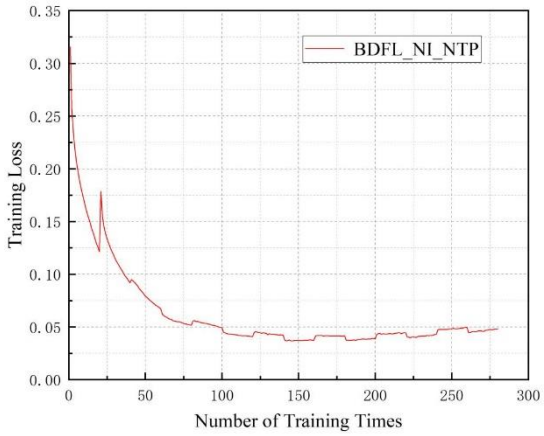


**Figure 2.** Loss values of the validation data during the training process.

Next, we compare the migration latency, the energy consumption, the punishment of refused SFC requests, and the total cost of these algorithms during the prediction process.

Figure 3 illustrates the average migration latency of the three algorithms under the same environment. The latency value is an important metric in the migration process, which shows the migration cost in each time slice. It can be observed from the figure that the VNF migration latency value of the BDFL_NI_NTP algorithm is the lowest. Conversely, the VNF migration latency value of the BlockFL algorithm is the highest. BlockFL increases by 57.06% compared to BDFL_NI_NTP; FL increases by 34.27% compared to BDFL_NI_NTP.

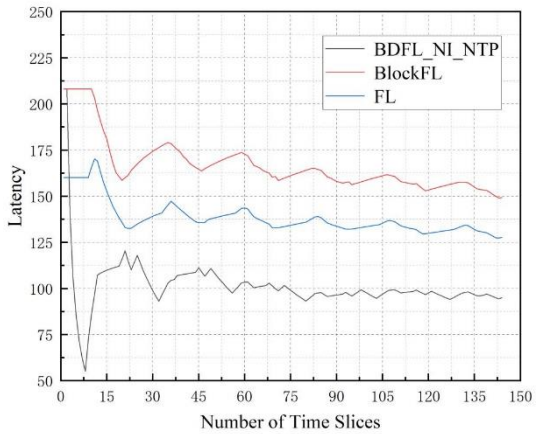


**Figure 3.** Migration latency during the prediction process.

Figure 4 depicts the average energy consumption of the three algorithms under the same environment. FL performs best in saving energy, but the differences among the three are not significant. FL increases by 0.08% compared to BDFL_NI_NTP; BlockFL increases by 0.23% compared to BDFL_NI_NTP.

 doi:10.20944/preprints202504.2606.v1



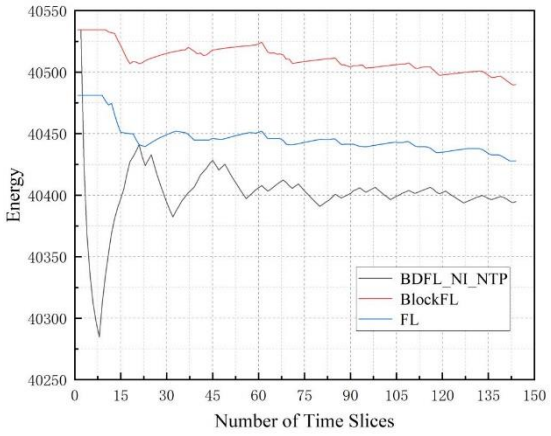


**Figure 4.** Energy consumption during the prediction process.

Figure 5 presents the total punishment for denying SFC requests of algorithms when the time slices number increases. We can observe that FL achieves the lowest. BlockFL increases by 22.35% compared to BDFL_NI_NTP; FL decreases by 39.53% compared to BDFL_NI_NTP.

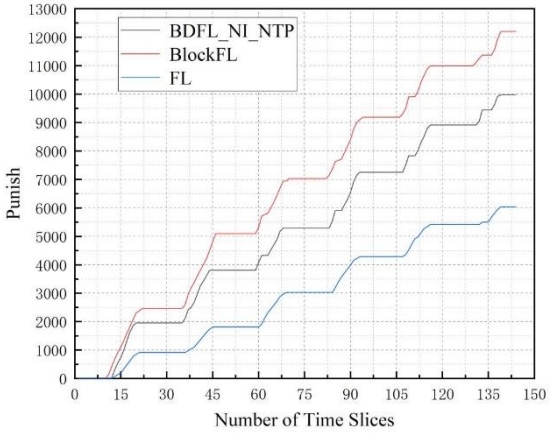


**Figure 5.** The punishment for SFC requests denying during the prediction process.

Figure 6 presents the average cost of the three algorithms when the time slices number increases. Through the above comparison, we find that BDFL_NI_NTP results in better performance. FL increases by 26.30% compared to BDFL_NI_NTP; BlockFL increases by 46.38% compared to BDFL_NI_NTP.

 doi:10.20944/preprints202504.2606.v1



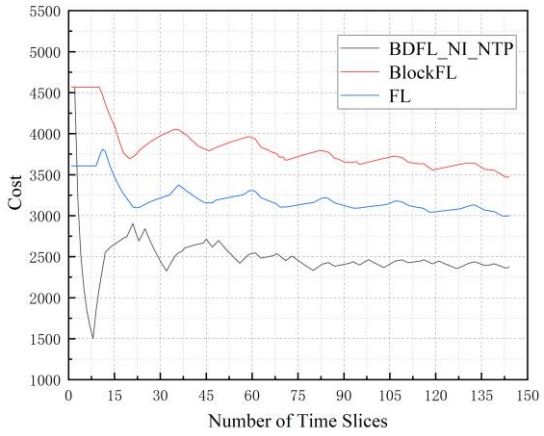


**Figure 6.** The cost during the prediction process.

Figure 7 presents the average cost of the three algorithms when the time slices number increases. Through the above comparison, we find that BDFL_NI_NTP results in the best performance. FL increases by 85.02% compared to BDFL_NI_NTP; BlockFL increases by 174.79% compared to BDFL_NI_NTP.

Finally, by varying the number of malicious nodes, the prediction results of the three algorithms are compared in terms of loss values and total costs, etc.

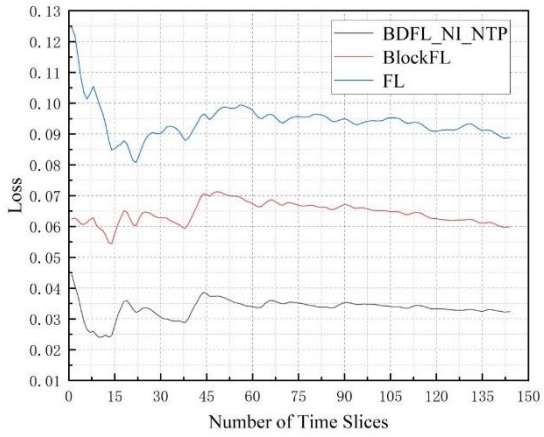


**Figure 7.** The loss during the prediction process.

As shown in Figures 8 and 9, as the proportion of malicious nodes increases, the migration cost and prediction loss value of the baseline algorithm's prediction results increase, while our scheme BDFL_NI_NTP shows little change in terms of cost and prediction loss value.

 doi:10.20944/preprints202504.2606.v1



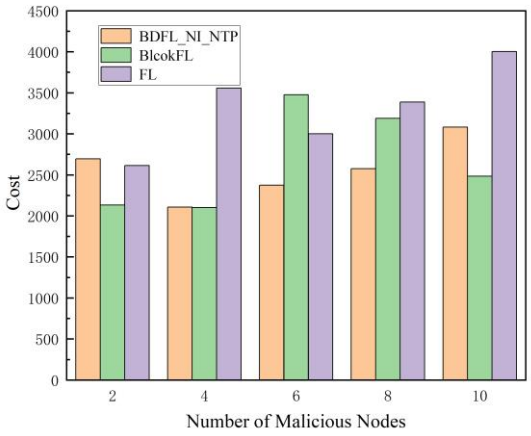


**Figure 8.** Cost value under different malicious nodes.

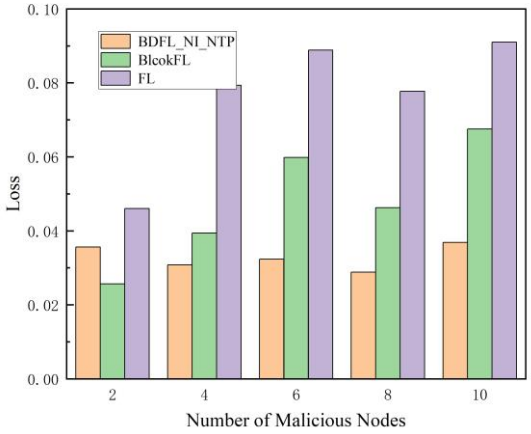


**Figure 9.** Loss value under different malicious nodes.

## 6. Conclusion

To address the challenges of cross-vendor collaboration, data privacy, and security in dynamic network traffic prediction within NFV environments, we propose an innovative solution integrating blockchain and decentralized federated learning with a node incentive mechanism. By dynamically reassigning node roles (e.g., Aggregator, Worker, Evaluator, and Residual nodes) to enable adaptive switching, the framework enhances prediction stability in node failure scenarios compared to traditional static architectures. The blockchain based decentralized federated learning architecture reduces VNF migration costs in experimental tests compared to conventional federated learning when there exist malicious nodes, because the adaptive dynamic role switching mechanism eliminates the impact of malicious nodes on global model aggregation.

This method provides a robust prediction framework for intelligent NFV operations, offering practical value for dynamic resource management in scenarios such as 5G network slicing and edge computing.

**Author Contributions:** Conceptualization, Y.H. and B.L.; methodology, Y.H.; software, Y.H.; validation, B.L.; formal analysis, Y.H. and B.L.; resources, B.L.; writing—original draft preparation, Y.H.; writing—review and editing, B.L.; visualization, B.L.; supervision, J.L.; project administration, L.J.; funding acquisition, Y.H. All authors have read and agreed to the published version of the manuscript.

 doi:10.20944/preprints202504.2606.v1



**Funding:** This work was partially supported by the Henan Provincial Department of Science and Technology Program, grant number 242102210204.

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

**Acknowledgments:** We extend our heartfelt appreciation to our esteemed colleagues at the university for their unwavering support and invaluable insights throughout the research process. We also express our sincere gratitude to the editor and the anonymous reviewers for their diligent review and constructive suggestions, which greatly contributed to the enhancement of this work.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Q. Liu et al., "Deep reinforcement learning for resource demand prediction and virtual function network migration in digital twin network," *IEEE Internet Things J.*, vol. 10, no. 21, pp. 19102–19116, Nov. 2023, doi: 10.1109/JIOT.2023.3281678.
2. J. Cai et al., "Privacy-preserving deployment mechanism for service function chains across multiple domains," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 1, pp. 1241–1256, Feb. 2024, doi: 10.1109/TNSM.2023.3311587.
3. P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021, doi: 10.1561/2200000083.
4. Z. Li et al., "Byzantine resistant secure blockchained federated learning at the edge," *IEEE Netw.*, vol. 35, no. 4, pp. 295–301, Jul. 2021, doi: 10.1109/MNET.011.2000604.
5. Q. Li et al., "AdaFL: Adaptive client selection and dynamic contribution evaluation for efficient federated learning," in *Proc. IEEE ICASSP*, Seoul, Korea, 2024, pp. 6645–6649, doi: 10.1109/ICASSP48485.2024.10447356.
6. Y. Hou, L. Zhao, and H. Lu, "Fuzzy neural network optimization and network traffic forecasting based on improved differential evolution," *Future Gener. Comput. Syst.*, vol. 81, pp. 425–432, Apr. 2018, doi: 10.1016/j.future.2017.08.041.
7. N. Ramakrishnan and T. Soni, "Network traffic prediction using recurrent neural networks," in *Proc. IEEE ICMLA*, Orlando, FL, USA, 2018, pp. 187–193, doi: 10.1109/ICMLA.2018.00035.
8. C.-W. Huang, C.-T. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *Proc. IEEE PIMRC*, Montreal, QC, Canada, 2017, pp. 1–6, doi: 10.1109/PIMRC.2017.8292737.
9. C. Pan et al., "DC-STGCN: Dual-channel based graph convolutional networks for network traffic forecasting," *Electronics*, vol. 10, no. 9, p. 1014, May 2021, doi: 10.3390/electronics10091014.
10. L. Huo et al., "A blockchain-based security traffic measurement approach to software defined networking," *Mobile Netw. Appl.*, vol. 26, pp. 586–596, Jun. 2021, doi: 10.1007/s11036-019-01420-6.
11. Y. Qi et al., "Privacy-preserving blockchain-based federated learning for traffic flow prediction," *Future Gener. Comput. Syst.*, vol. 117, pp. 328–337, Apr. 2021, doi: 10.1016/j.future.2020.12.003.
12. H. Guo et al., "B2SFL: A bi-level blockchained architecture for secure federated learning-based traffic prediction," *IEEE Trans. Serv. Comput.*, vol. 16, no. 6, pp. 4360–4374, Nov. 2023, doi: 10.1109/TSC.2023.3318990.
13. V. Kurri, V. Raja, and P. Prakasam, "Cellular traffic prediction on blockchain-based mobile networks using LSTM model in 4G LTE network," *Peer-to-Peer Netw. Appl.*, vol. 14, pp. 1088–1105, May 2021, doi: 10.1007/s12083-021-01085-7.
14. L. Feng et al., "BAFL: A blockchain-based asynchronous federated learning framework," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1092–1103, May 2022, doi: 10.1109/TC.2021.3072033.
15. J. Li et al., "Blockchain assisted decentralized federated learning (BLADE-FL): Performance analysis and resource allocation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2401–2415, Oct. 2022, doi: 10.1109/TPDS.2021.3138848.

 doi:10.20944/preprints202504.2606.v1



16. D. Połap, G. Srivastava, and K. Yu, "Agent architecture of an intelligent medical system based on federated learning and blockchain technology," *J. Inf. Secur. Appl.*, vol. 58, p. 102748, Jun. 2021, doi: 10.1016/j.jisa.2021.102748.
17. L. Cui, X. Su, and Y. Zhou, "A fast blockchain-based federated learning framework with compressed communications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3358–3372, Dec. 2022, doi: 10.1109/JSAC.2022.3213345.
18. Y. Ren et al., "BPFL: Blockchain-based privacy-preserving federated learning against poisoning attack," *Inf. Sci.*, vol. 665, p. 120377, Mar. 2024, doi: 10.1016/j.ins.2024.120377.
19. H. Kasyap, A. Manna, and S. Tripathy, "An efficient blockchain assisted reputation aware decentralized federated learning framework," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 2771–2782, Sep. 2023, doi: 10.1109/TNSM.2022.3231283.
20. S. Qiao et al., "LBFL: A lightweight blockchain-based federated learning framework with proof-of-contribution committee consensus," *IEEE Trans. Big Data*, 2024, doi: 10.1109/TBDATA.2024.3481952.
21. C. Che et al., "A decentralized federated learning framework via committee mechanism with convergence guarantee," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4783–4800, Dec. 2022, doi: 10.1109/TPDS.2022.3202887.
22. Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Jan. 2019, doi: 10.1145/3298981.
23. H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020, doi: 10.1109/LCOMM.2019.2921755.