The Scientific Temper (2025) Vol. 16 (6): 4391-4402

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2025.16.6.12

https://scientifictemper.com/

RESEARCH ARTICLE

Distributed SDN control for IoT networks: A federated meta reinforcement learning solution for load balancing

A. Sandanasamy[1*], P. Joseph Charles[2]

Abstract

The growth of Internet of Things devices and their uses have introduced ample challenges in handling dynamic and heterogeneous traffic patterns. This also has affected the area of Software Defined Networking (SDN). The key parameters like scalability, latency and resilience are the concerns in centralized SDN approach, especially in the case of large-scale IoT deployments. This research introduces a new method, Distributed SDN Control for IoT networks: A Federated Meta Reinforcement Learning Solution for Load Balancing. This method combines Federated Learning (FL) with the key features of Meta Reinforcement Learning (Meta-RL) to enable intelligent and privacy preserving load balancing across distributed SDN controllers. The system functions in two phases. In the first phase, traffic distribution models across are trained with FL without sharing raw data. Security is added to this by differential privacy and Byzantine-resilient aggregation. In the second phase, fast adaptation to non-stationary traffic patterns is achieved using Meta-Learning and Proximal Policy Optimization (PPO). The performance evaluations show that the proposed method improves load balancing efficiency. It also reduces the response time and maintains resilience in dynamic traffics.

Keywords: Internet of Things, Load Balancing, SDN – IoT, QoS, Software Defined Networking, Proximal Policy Optimization.

Introduction

The modern digital ecosystems are deeply affected by the rapid growth of Internet of Things (IoT). Interconnected with billions of devices, it continuously produces and process vast amount of data (Allioui *et al*., 2023). Because of these advancements, immense opportunities have been created in automation, monitoring and real time decision making. And the fields include health care, smart cities, agriculture and etc., (Angelpreethi *et al*., 2016). This also affects the traditional network especially the parameters such as scalability, adaptability, latency and load management (Sinduja *et al*., 2025).

Software Defined Networking introduces a change in the network management with flexibility and programmability. It is done by decoupling the functionalities of control plane from data plane. The centralized control gives global view to the network administrator. It makes easy to optimize routing, enforce policies and manage traffic. But in the case of geographically distributed IoT networks, the centralized approach becomes a blockage. It can lead to latency, poor fault tolerant and single point of failure while handling dynamic and high throughput IoT network (Kazmi *et al*., 2023). To solve this issue, Distributed SDN is introduced. Multiple controllers are deployed to manage the network responsibilities. As the result, scalability, fault tolerance and responsiveness are improved. But it suffers in load balancing, especially in the unpredictable and heterogeneous IoT environment (Mathanraj *et al*., 2024). As a solution, intelligent decision making system is need for optimal resource allocation, low latency and improving other network parameters (Bannour *et al*., 2018).

To face these challenges, Machine Learning based solutions tend to be serving better in dynamic traffic management. While considering all the machine learning approaches, Reinforcement Learning (RL) servers better in optimizing network resource allocation through continuous

[1]Department of Computer Applications, Bishop Heber College, Affiliated to Bharathidasan University, Tiruchirappalli, Tamil Nadu, India.

[2]Department of Computer Science, St. Joseph's College, Affiliated to Bharathidasan University, Tiruchirappalli, Tamil Nadu, India.

***Corresponding Author:** A. Sandanasamy, Department of Computer Applications, Bishop Heber College, Affiliated to Bharathidasan University, Tiruchirappalli, Tamil Nadu, India, E-Mail: sandanasamy.ca@bhc.edu.in

How to cite this article: Sandanasamy, A., Charles, P.J. (2025). Distributed SDN control for IoT networks: A federated meta reinforcement learning solution for load balancing. The Scientific Temper, **16**(6):4391-4402.

Doi: 10.58414/SCIENTIFICTEMPER.2025.16.6.12

Source of support: Nil

Conflict of interest: None.

Received: 18/06/2025 **Accepted:** 20/06/2025 **Published**: 30/06/2025

learning from the environment. Reinforcement Learning depends on centralized training. It creates a concern for privacy, scalability, and resistance to security attacks, especially in distributed network (Zhuang *et al*., 2023). Here, Federated Learning the solution by offering decentralized training framework, where collaborative training occurs with multiple SDN controllers without exchanging the raw data. Thus it preserves privacy and scalability. But it lacks to manage against malicious participant such as Byzantine attacker (Raza *et al*., 2024).

To handle all these challenges, this research proposes a holistic solution to the SDN IoT decentralized limitations. The proposed method is "Distributed SDN control for IoT Networks: A Federated Meta-Reinforcement Learning Solution for Load Balancing". The proposed Federated Meta-Reinforcement learning based Load Balancing (FMRLB) has the following components. Federated Learning phase is used for privacy preserving model training. Meta Reinforcement Learning is used for rapid adaptation to localized traffic conditions (Kazmi *et al*.,2023). Stable and efficient decision making is achieved by Proximal Policy Optimization. Long Short Term Memory provides proactive traffic prediction. Byzantine filtering is used for protecting the federated model against misleading updates (Chien *et al*., 2024). This integrated approach makes SDN controller to provide intelligent, privacy aware load balancing decisions. The design produces low latency, faster adaptability to dynamic network and resistance to threats.

Related Work

The authors (Zormati and Lakhlef, 2023) proposed a distributed intelligent network system. They suggested five layer approaches with Application, Control, Virtualization Learning and infrastructure layers. These layers collaboratively manage the complexity and scalability issues in IoT networks. Here SDN provides centralized programmability. NFV provides flexible and on demand deployment of network. In this proposed model, distributed hierarchical SDN control is used with a root controller. Federated Learning provides privacy preserving decentralized capacities. The proposed method shows how distributed intelligence reduces communication overhead through federated learning. But this does not have intelligent load balancing for dynamic load balancing.

A federated learning framework tailored for Software-Defined Networking (SDN) environments was introduced by (Tran and Tran, 2024). This research addresses the critical issues in distributed and heterogeneous network. It aims to provide solution through decentralized model training across multiple SDN controllers. Instead of sharing the raw data, local model are trained in each SDN controller maintaining privacy. Then they are aggregated for collective learning. The proposed system also addresses the issues of communication overhead and system heterogeneity. This is done by optimizing the aggregation of local models. Here aggregation optimization is achieved but it is not adaptive to traffic patterns.

The authors in (Ma *et al*., 2022) make a survey to study the potential application of Federated Learning with Software defined networking. This study explores the centralized control capabilities of SDN and decentralized learning mechanisms of Federated Learning. The study indicates several key challenges that occur at the interactions of these two technologies. One challenge is to manage the need for effective incentive system to motivate data and resource sharing. Another challenge is the privacy preserving and security during the model training and exchange. Aggregating heterogeneous models in diverse networks stands as another key challenge. As the authors review these challenges, they also suggest potential enhancements. They are privacy preserving mechanisms, adaptive aggregation system and incentive models mainly for SDN environments.

A security based model based on Deep Federated Learning was introduced by the author (Albogami, 2025) . This research focuses on improving security aspects in Internet of Things. This system uses a Federated Hybrid Deep Belief Network to analyse temporal data generated by edge sensors in IoT networks. This mechanism of processing data locally enables privacy preserving machine learning. The pre processing steps of data normalization and feature selection is done using Golden Jackal Optimization. This utilizes the Dung Beetle Optimizer for fine tuning hyper parameters. It lacks SDN integration and network level controlling mechanism.

The authors (Mahmod *et al*., 2025) proposed a Software Defined Networking enhanced framework to optimize client selection in federated learning. This research aims to improve client selection in IoT and edge computing environments. Traditional methods used to select clients randomly for training data. This caused inefficiencies because of variances in capabilities of devices and network conditions. To manage this, the proposed method used the advantage of SDN to monitor the network metrics like bandwidth, latency, energy levels and client specific factors like computational power and data quality globally. Because of this, the system was able to intelligently select the clients with stable connectivity, high quality dataset and sufficient processing capabilities. Mininet emulator was used to evaluate this proposed method. The result shows that there is a significant level of reduction in communication overhead. This work focuses mainly on training phase and no real time operations.

A multi agent solution for dispatching the requests in a distributed SDN environment was introduced by the authors (Huang *et al*., 2023). This research work focuses on requests dispatching in distributed software defined networking in control planes. Normally, SDN architectures use centralized

controllers. These raises the issue of scalability and reliability in large networks. As an alternate, when distributed control planes are used, it introduces complexities in load balancing and request dispatching. The authors proposed solution to the above said issues. They developed multiagent deep reinforcement learning. It makes the SDN switch to make dispatching decisions by itself without any global network state information. This is designed to manage real time network changing conditions and varying number of controllers. There is no global coordination in this approach.

Though there are advancements in integrating federated learning, software defined networking and deep reinforcement learning, the existing approaches do not have a unified, adaptive and privacy preserving mechanism. They lack comprehensive integration of predictive traffic modelling, adaptive decision making and secure distributed co-ordination. This creates a gap for developing a federated approach with privacy and trust oriented mechanism to manage real time network conditions.

System Model

The architectural diagram presents the design of Federated Meta-Reinforcement Learning based Load Balancing (FMRLB). Figure 1 demonstrates the entire architecture of the system. This system provides a load balancing oriented approach for adaptive traffic management in SDN-

IoT environments. This has three main core layers. They are application layer (high level), control layer (decision making) and data layer (execution level). They are interconnected through Northband and Southband Interfaces (Madani *et al*., 2023). The application plane manages the entire load balancing scheme. It initiates model

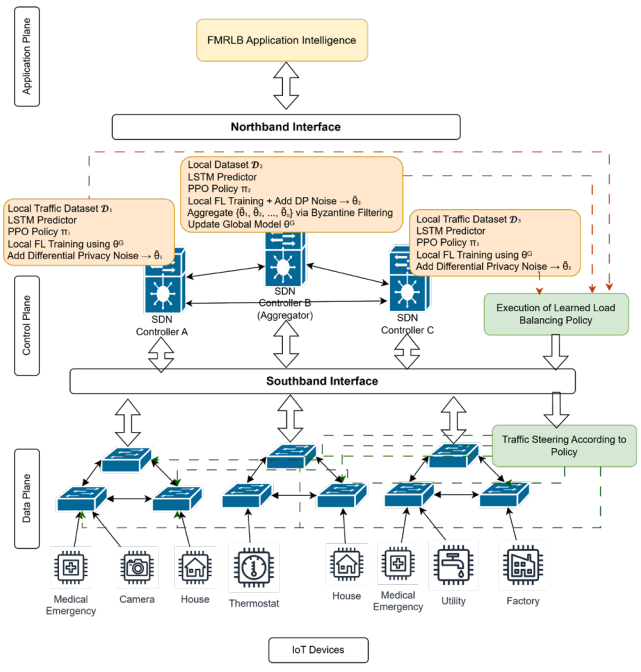

Figure 1: System architectural diagram

training, updates the global coordinates and distributes the optimal policy to all SDN controllers. This layer can be considered as the brain of the entire system. It guides all learning and load balancing decision making process. At control plane, each SDN controller manages a local traffic dataset (D_i) and predicts future data traffic using LSTM predictor. It also uses Proximal Policy Optimization (PPO) for leaning efficient load balancing policy δ_i (Zhou *et al*., 2024).

Differential Privacy noise is incorporated with training data so that the sensitive data are safeguarded. Each controller submits their noisy updates $\tilde{\theta}_i$ for Byzantine-robust aggregation (Wang *et al*., 2024). This help in forming a global model θ^G. This mechanism ensures that the local data privacy is concentrated while participating in collaborative load balancing. At the data plane the SDN switches directly interact with the IoT devices. The switches learn the load balancing schemes from PPO and executes routing in real time. This helps in reducing link congestion, optimizing throughput, enhancing quality of service in IoT network. The multilayer architecture enables privacy preserving, intelligent and decentralized load balancing. It supports continuous learning with policy adaptation in response to dynamic network conditions.

The Figure 2 illustrates the complete workflow of the FMRLM. The adaptive load balancing mechanism begins with data acquisition of network metrics like bandwidth, delay and packet drops, CPU usage, RAM utilization, queue length, and link utilization. These data are converted as a local dataset for each controller. Each controller performs the local Federated Learning training. It adds differential privacy noise (DP) and also applies Byzantine filtering. Then a convergence check is conducted. If the convergence is reached, meta reinforcement learning is applied to generalize across varying networks.

If the convergence is not reached then additional rounds are executed. The learned policy also undergoes a performance check. If the policy is found ineffective, then further learning is continued. When the policy is effective then Proximal Policy Optimization is used to refine the load balancing policy. An LSTM based traffic predictor anticipates the future traffic. This leads to proactive load balancing. Updated metrics are collected and feedback loop is executed for continuous learning and adaptive decision making.

Methodology

The proposed Federated Mete-Reinforcement learning based Load Balancing aims to provide a privacy driven intelligent decision making approach in handling adaptive traffic. It integrates several modules for performing the load balancing in the dynamic networks. They are Federated Leaning, Differential Privacy, Byzantine Resilient Aggregation, Meta-Reinforcement Learning, Long Short

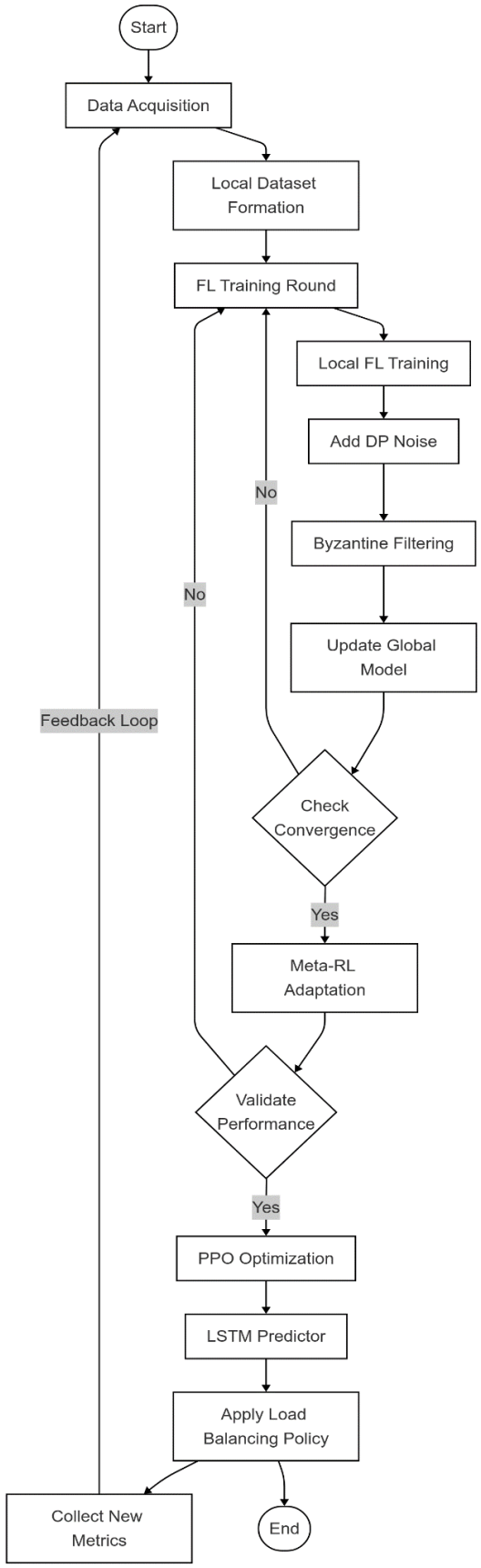

Figure 2: Flow Diagram of the System

Term Memory based traffic prediction and Proximal Policy Optimization. Each module performs its tasks in the distributed SDN architecture.

The IoT network traffic data are collected from each SDN controller $c_k \in \{c_1, c_2, \ldots, c_N\}$. The collected data are used construct the feature vector $x \in R^d$ with key metrics. The metrics are bandwidth consumption, delay, packet drop rate, CPU usage, RAM utilization, queue length, and link utilization. These features represent resource status as well as network performance. It acts as the basis for effective load balancing. The variable 'y' is the target variable. It corresponds to the traffic load or congestion level for each sample data. Each controller constructs a rich local dataset as the time progresses. It is represented as

$$D_k = \{(x_i, y_i)\}_{i=1}^{m}$$

It contains m input output pairs for local model training. This data set is used by each controller to train a local prediction model θ_k. Using this, the loss function $L_k(\theta_k)$ is calculated.

$$L_k(\theta_k) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\theta_k}(x_i) - y_i\right)^2 + \frac{\mu}{2}\| \theta_k - \theta^G \|^2$$

In the loss function, $f_{\theta_k}(x_i)$ is the predicted traffic load, θ^G is the current global model. μ is the regularization parameter. The training phase is done using all the key features notably bandwidth, delay, drop, CPU, RAM, queue and link utilization to ensure accurate prediction and effective optimization.

Each controller perturbs its learned model with Gaussian noise to preserve privacy and security.

$$\tilde{\theta}_k = \theta_k + N(0, \sigma^2) \text{ where } \sigma = \sqrt{\frac{2\ln(1.25/\delta)}{\dot{o}}}$$

In the above noise construction, σ denotes the noise scale. $\dot{o}$ is the privacy budge. δ is the confidence level. A stronger privacy is enforced by using smaller $\dot{o}$ at the cost of greater noise. When all noisy updates $\tilde{\theta}_k$ are received, a Byzantine Resilient Aggregation (Krum) is applied. It mitigates the adversarial behaviour. The Krum aggregator identifies the model with more similarity to the majority of the updates filtering outliers.

$$\text{Krum}\left(\{\tilde{\theta}_k\}\right) = \arg\min_{\theta_k} \sum_{i \neq j} I\left(\| \theta_i - \theta_j \|^2 > \tau\right) \text{ where } \tau = \mu + 2\sigma$$

This mechanism makes sure that no malicious controllers distort the global model θ^G. And thus it enhances security

and reliability. Upon updation of the global model, it is disseminated to all the controllers. Meta Learning is incorporated into FMRLB system to enable rapid adaptation to dynamic traffic with few gradient steps. Each controller using Meta RL adapts the global model to its local network conditions. For a given controller c_k and its local task T_k, the adapted model is computed.

$$\theta_k^{'} = \theta^G - \alpha \nabla_\theta L_{T_k}\left(\theta^G\right)$$

The meta-learning rate is represented as α in the above adapted model. The same network feature: bandwidth, delay, drop rate, CPU usage, RAM usage, queue length, and link utilization are used in the loss function L_{T_k}. The model adapts quickly to changing local traffic using the meta learning. In the next step, the personalized model $\theta_k^{'}$ is used for initializing the PPO based load balancing policy π_k. The PPO is applied to improve the clipped surrogate objective.

$$\max_\pi E_t\left[\min\left(r_t\left(\theta\right)\hat{A}_t, \tilde{r}_t\left(\theta\right)\hat{A}_t\right)\right]$$

In the above objective, the ratio of new policy and old policy probabilities for action a_t in state s_t is represented as $r_t(\theta)$. $\tilde{r}_t(\theta)$ is the clipped ratio. It ensures $r_t(\theta)$ stays within $[1-\epsilon, 1+\epsilon]$ by using a lower bound and upper bound $\hat{A}_t$ is the estimated advantage. The state vector s_t construction is mentioned below, s_t is the state vector and maintains

$$\left[h_t, \mathrm{CPU}_k(t), \mathrm{RAM}_k(t), \mathrm{BW}_k(t), \mathrm{Delay}_k(t), \mathrm{Drop}_k(t), \mathrm{Queue}_k(t), \mathrm{LU}_k(t)\right]$$

State vector consists of h_t, the hidden state from the LSTM predictor, real time metrics: CPU and RAM usage, bandwidth consumption, end-to-end delay, packet drop rate, queue length, and link utilization (LU) at time t for controller k. The complete network context is represented in the decision making process.

The LSTM operates on the past traffic load measurements and predicts the future congestion trends and provides proactive decision making. The internal computation of the LSTM model are

$\mathrm{f_t} = \sigma\left(\mathrm{W_f} * \left[\mathrm{h_{t-1}}, \mathrm{L(t)}\right] + \mathrm{bias_f}\right)$ represents forget gate

$\mathrm{i_t} = \sigma\left(\mathrm{W_i} * \left[\mathrm{h_{t-1}}, \mathrm{L(t)}\right] + \mathrm{bias_i}\right)$ represents input gate

$\tilde{\mathrm{C}}_\mathrm{t} = \tanh\left(\mathrm{W_C} * \left[\mathrm{h_{t-1}}, \mathrm{L(t)}\right] + \mathrm{bias_C}\right)$ represents candidate cell state

$\mathrm{C_t} = \mathrm{f_t} \odot \mathrm{C_{t-1}} + \mathrm{i_t} \odot \tilde{\mathrm{C}}_\mathrm{t}$ represents cell state update

$\mathrm{o_t} = \sigma\left(\mathrm{W_o} * \left[\mathrm{h_{t-1}}, \mathrm{L(t)}\right] + \mathrm{bias_o}\right)$ represents output gate

$h_t = o_t \odot \tanh\left(C_t\right)$ represents hidden state

σ is the activation function. $L(t)$ is the current traffic load computed from the same set of features. The $\hat{L}(t+1)$ output servers as the next time step's load, providing proactive traffic balancing. W_f, W_i, W_C, W_o are weight matrices The end to end data flow are continued in rounds. The real time metrics are collected. The differentially private are aggregated with Byzantine resistance. Personalized policies are adapted from meta reinforcement learning. And the future loads are predicted using LSTM.

The federated learning component has the theoretical guarantee of convergence under DP and adversarial conditions. The global model θ_T^G converges towards the optimal solution θ^* with the following bound

$$\| \theta_T^G - \theta^* \| \leq \left(1-\eta\mu\right)^T \| \theta_0^G - \theta^* \| + \frac{2G\sigma}{\mu\sqrt{N}} + O(\delta)$$

Here, η is the learning rate. μ is the strong convexity constant. To ensure stability, N is the number of controllers. G is the gradient size. σ is the amount of variance. And δ is the privacy budget.

The Byzantine resilient aggregator maintains robustness against malicious controller during federated learning. This ensures that the global model θ^G remains close to θ^* with the bounded error of

$$\| \theta^G - \theta^* \| \leq O\left(f / N\right)$$

f represents the malfunctioning controllers. N represents total controllers. By incorporating Byzantine resilient, the FMRLB handles adversarial behaviour without compromising the accuracy of load balancing decisions. This entire methodology integrated using federated learning, differential privacy, Byzantine resilience, meta reinforcement and LSTM provides efficient load balancing with security features.

Algorithm: Federated Meta Reinforcement Learning with LS TM and Differential Privacy for Adaptive Load Balancing

Input: { N, T , $\mathcal{D}_k$, θ^0, ε = 1.0, δ = 1e-5,α , μ, τ , m, γ = 0.99, λ, η, Δ, MaxRounds }

Output:{ Final global model (θ_G), set of trained policies $(\pi_1, \pi_2, \ldots \pi_N)$ }

Initialization

The global parameter is initialized as $\theta^G \leftarrow \theta^0$

Training Procedure

The training phase : till the convergence reaches or the number of rounds is reached.

Set $t \leftarrow 0$

While the global model exceeds η and t < MaxRounds do

Local Task Sampling :

Each controller c_k (k=1 to N} performs local

training independently

Create a task $\mathcal{T}_k$ by selecting a temporal or random subset from $\mathcal{D}_k$

Meta Training Phase

Set $\theta_k = \theta^G$

Perform S times local iterations :

Sample mini batch B_k from $\mathcal{T}_k$

Find the gradient

$g = \nabla\_\theta\ [(1/m)\ \Sigma_i\ (f_k(x_i) - y_i)^2 + (\mu/2)\|\theta - \theta^G\|^2]$

Gradient Clipping for stability

$$g \leftarrow g \cdot \min\left(1, \frac{\Delta}{\| g \|}\right)$$

Each controller is applied local update with scaled gradient

$$\theta_k = (\theta_k - \alpha \cdot g)$$

Privacy Preservation

Add Differential Privacy noise to θ_k with σ calculated from ε and δ

Byzantine Aggregation (Multi-Krum):

Calculate the distance between ever pair of $\tilde{\theta}_k$ updates. Select m models with consistency Revise the global model:

$$\theta^G = (1/m)\ \Sigma_{\{\tilde{\theta}_k \in S\}}\ \tilde{\theta}_k$$

If the select model is not sufficient, do median based approach

Calculte θ_{median} for all $\tilde{\theta}_k$ vectors

Weigh Assignment $w_k = 1 / (\|\tilde{\theta}_k - \theta_{median}\| + \varepsilon)$

Updating Consolidation

$$w_k \leftarrow \frac{w_k}{\sum w_k}, \theta^G \leftarrow \sum w_k \cdot \tilde{\theta}_k$$

Meta Adaptation and LSTM, PPO Training Per Controller:

(Each controller getting specialized with meta adaptation and reinforcement learning)

Single step adaptation using current global model

$$\theta_{k'} \leftarrow \theta^G - \alpha \nabla_\theta L_{T_k}\left(\theta^G\right)$$

Pre-fill LSTM memory with historical data $L_{\{t-T:t\}}$ from $\mathcal{D}_k$:

for step = 1 to 100:

h_t = LSTM.update($L_{\{t-T:t\}}$)

Reinforcement Learning

At each point in time t, the following actions and rewards are done

Perform state construction with $s_t = [h_t, CPU_k(t), RAM_k(t), BW_k(t), Delay_k(t), Drop_k(t), Queue_k(t), LU_k(t)]$

Determine a_t from s_t with π_k

Record r_t and $s_{\{t+1\}}$

PPO Policy Update

$$r_t(\theta) = \frac{\pi_k(a_t \mid s_t)}{\pi_k^{\text{old}}(a_t \mid s_t)}$$

A_t with GAE(γ, λ)

$$\tilde{r}_t(\theta) = \max(1-\epsilon, \min(r_t(\theta), 1+\epsilon)$$

$$L^{CLIP}(\theta) = E_t\left[\min\left(r_t(\theta) A_t, \tilde{r}_t(\theta) \hat{A}_t\right)\right]$$

Return

Global model $\{\theta^G\}$, local policies $\{\pi_1, \pi_2, ..., \pi_N\}$

Experimental Setup

The proposed FMRLB was evaluated in a carefully designed IoT-SDN environment using Mininet (Bagde *et al.*, 2024). The network structure is designed with 50 resource constrained IoT devices. These devices include various sensors and actuators. These devices were grouped into five clusters. Each cluster was managed by one of five P4 programmable switches (Miguel-Alonso, 2023). In the control plane, five controller ONOS instances (Rahim *et al.*, 2024) were implemented as controller instances, running on separate virtual machines. The adversarial conditions were introduced using 10% of IoT devices were configured as Byzantine nodes. Thus it was made to inject faulty federated learning updates and UDP attacks. Three traffic profiles were used namely periodic sensor data, bursty video streams and adversarial traffic. The variables used in the algorithm are listed in the Table 1.

The network conditions were calibrated with bandwidth throttling to 10Mbps per link using Linux tc. The latency of 10ms with 2ms variance is used for real time environment. This implementation was integrated with differential privacy ε=2.0, δ=10^{-5}. Local model updates with cosine similarity below 0.85 relative to the major cluster direction were discarded as potential malicious contributions. Krum-based Byzantine resilient aggregation with threshold τ=1.7 and LSTM predictors for traffic forecasting. Meta Reinforcement learning adaptation was done using PPO policy with clipped objectives ε=0.2. All the data collection was done using tool chain Mininet, ONOS, P4 switches. Logs were processed using python scripts and plotting is done.

Result Analysis

The effectiveness of FMRLB algorithm is compared against Vanilla Federated Leaning(Vanilla FL), Krum-only algorithm and Federated Learning integrated with Meta Reinforcement(FL+MetaRL). The throughput performance of different learning methods across 100 communication rounds are presented in Figure 3. DDoS and Byzantine attacks are introduced at specific intervals. This is highlighted with shaded region. The experiment shows the FMRLB has higher throughput. It also demonstrates faster recovery after attacks compared to other methods. FL+MetaRL has

Table 1: Variable Description

Variable	*Description*
N	Total number of SDN controllers.
T	The size of the time window.
D_k	Local traffic dataset stored at controller c_k.
θ^0	The initial version of the global model
ε	Privacy budget in differential privacy.
δ	Privacy loss parameter used in DP analysis.
α	Local model learning rate.
μ	Over fitting monitor..
τ	Distance threshold (identifying and filtering out malicious updates).
m	Number of selected model updates retained during the Multi-Krum aggregation step.
γ	Value used for finding importance of future reward
λ	Generalized Advantage Estimation parameter
η	Tolerance level for convergence in global model updates.
Δ	Clipping threshold.
θ^G	Global Model.
θ_k	Model parameters locally updated at controller c_k.
θ_k	Differentially private version of the locally updated model θ_k.
g	Local model gradient vector.
σ	Standard deviation used for noise in differential privacy, computed as: $\sqrt{(2\cdot\ln(1.25/\delta))}\cdot\Delta/\varepsilon$.
S	Set of reliable model updates selected during robust aggregation.
π_k	Policy model trained via PPO and deployed on controller c_k.
θ_k'	Adapted version of the global model fine-tuned for controller c_k.
T_i	A small task or batch randomly sampled from the local dataset D_k.
B_k	Mini-batch of data samples used during the local update step.
s_t	System state at time t, including hidden patterns and current network metrics.
h_t	Hidden state output from the LSTM capturing recent traffic behavior.
$CPU_k(t)$	CPU usage recorded at controller c_k at time t.
$RAM_k(t)$	Memory usage at controller c_k during time t.
$BW_k(t)$	Current bandwidth at time t..
$Delay_k(t)$	Measured packet delay at the given timestamp.
$Drop_k(t)$	Rate at which packets are dropped at controller c_k.
$Queue_k(t)$	Current length of the packet queue.
$LU_k(t)$	Link utilization observed at controller c_k.
a_t	Action chosen by the PPO policy at step t.
r_t	Reward received after executing action a_t.
A_t	Estimated advantage at time t.
$r_t(\theta)$	Ratio between the new and old policy probabilities in PPO.
$L^{CLIP}(\theta)$	Clipped function.

degradation during attack phases. Krum-only and Vanilla FL tend to have drop in the performance. They have slower recovery and lower overall throughput. This explains the effectiveness of FMRLB. And it also affirms robust and stable performance even under attacks.

Figure 4 demonstrates the latency analysis. The evaluation of FMRLB approach for the latency behaviour and compared with other federated learning approaches across 100 rounds. This was done under normal and adversarial conditions of DDos and Byzantine attacks. FMRLB performs better with

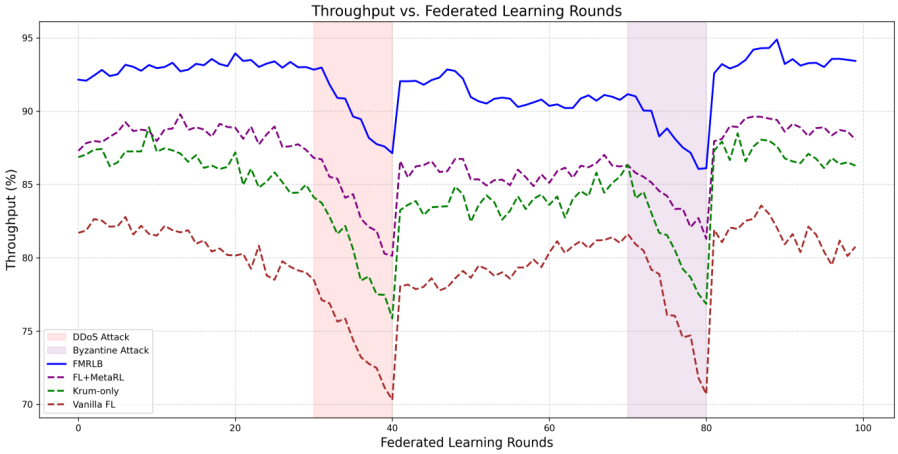

Figure 3: Throughput Analysis

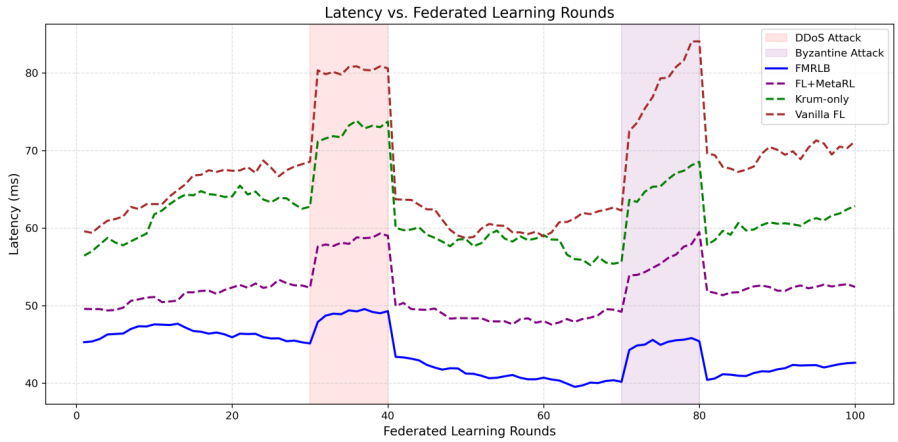

Figure 4: Latency Analysis

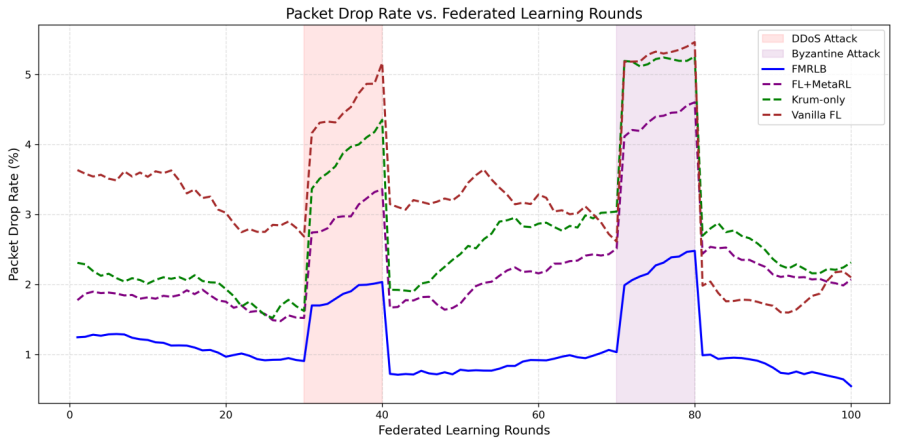

Figure 5: Packet Drop Rate Analysis

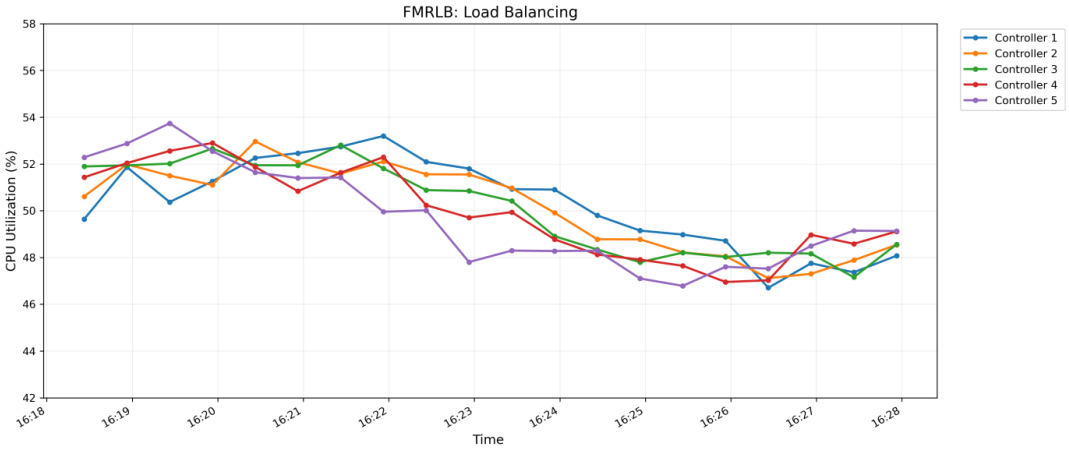

Figure 6: CPU Utilization Comparisons

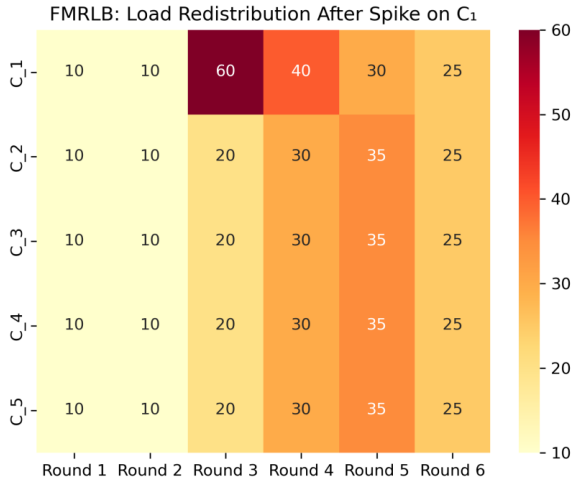

Figure 7: Controller Load Distribution

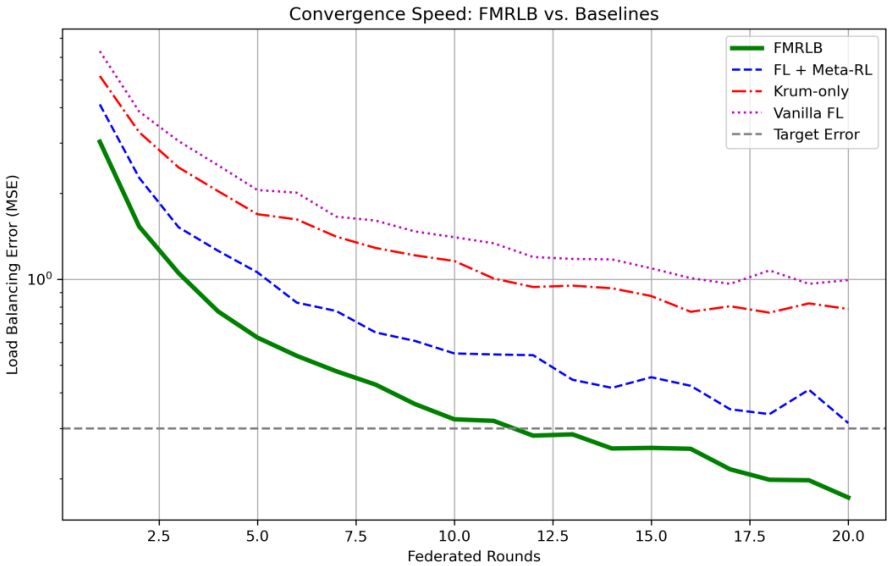

Figure 8: Learning Convergence Analysis

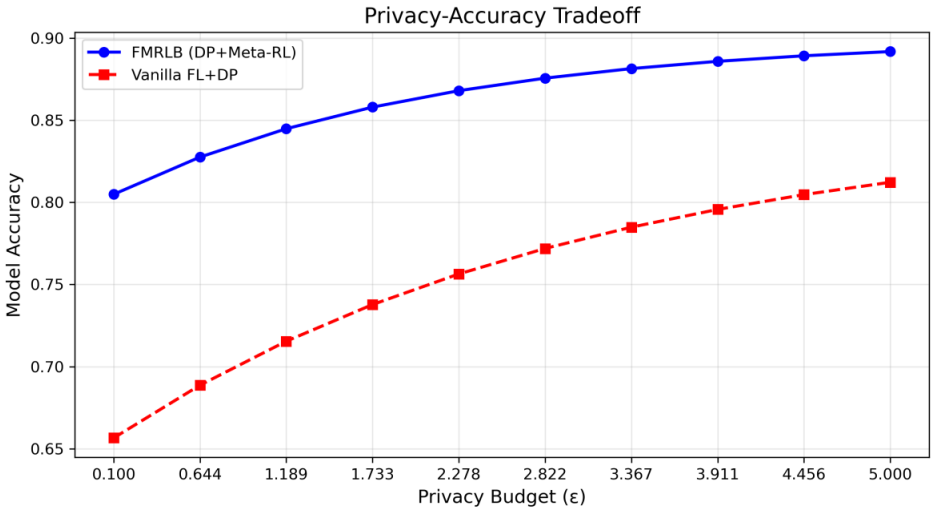

Figure 9: Privacy Preserving Comparison

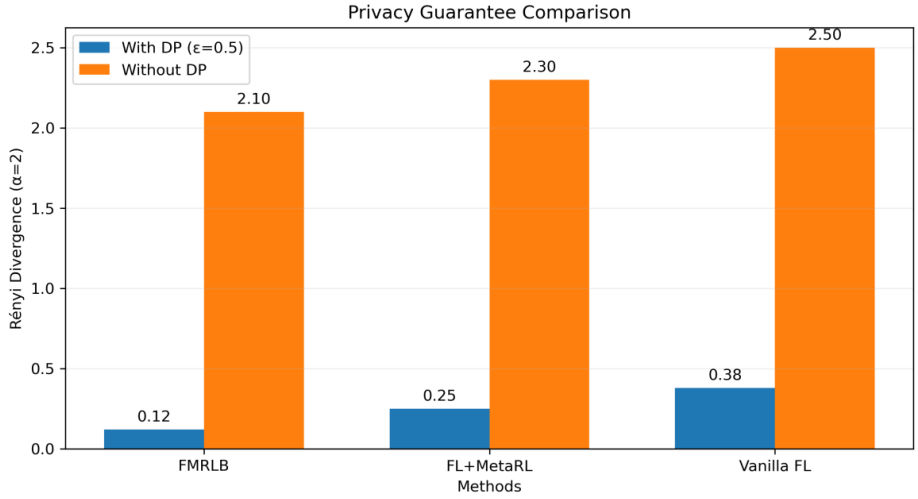

Figure 10: Privacy Preserving Analysis

lowest latency. It also shows rapid stabilization after attack periods. The latency of FL+MetaRL tends to be moderate but it creases during the attacks. Krum-only and Vanilla FL has higher latency. They also have slower recovery and high instability. These findings state that FMRLB performs better than other baseline algorithms even under the adversarial conditions.

Figure 5 shows the packet drop rate comparison of FMRLB with other baseline algorithms. These are evaluated under normal conditions and attack conditions with DDoS and Byzantine. FMRLB consistently performs with lowest packet drop rates. It also quickly stabilizes even under attacks. But , FL+MetaRL, Krum-only and Vanilla FL have high spikes indicating high packet drops. Vanilla FL shows highest and sustained packet loss. Once the attack period is completed, FMRLB gets back to the baseline performance level. This demonstrates the high resilience in managing communication reliability.

The Figure 6 presents the CPU utilization in percentage over time for five distributed controllers. The result communicates that the computations workload is managed efficiently. It depicts stable performance across all nodes. The CPU utilization level of all the controllers remain intact without any significant outliers. This consistency underlines the capacity in distributing the load without overutilization of any single controller. The absence of any spikes or fluctuations highlight the stability of the proposed approach. This highlights the suitability to adapt itself in

dynamic workload environments. These findings affirm the practical applicability of the proposed algorithm in real time load management.

Figure 7 shows the result of investigation done to find the dynamic load redistribution of the FMRLB approach in a simulated environment. A sudden load spike is generated on C1 and its performance is evaluated. The heatmap shows the load adjustments in six rounds among five nodes C1 to C5. At round1 , all nodes go through balanced load. During Round3, a spike is introduced on C1. Sensing the load, FMRLB distributes excess load across other nodes. Further rounds show that the system has efficient balancing with C2 to C5, reducing the strain on C1. By round 6, the stability is achieved. This indicates the FMRLB's ability to adapt load surges. The colour intensity from light yellow to dark red visualizes the degree of load at each node per round. This result affirms the proposed method's potentiality for use in dynamic and high demand environments

Figure 8 shows the convergence ability of FMRLB compared with baseline algorithms. The FMRLB exhibits faster reduction in Load balancing error measured in MSE across federated rounds. FMRLB reaches the target error threshold approximately in 11 to 12 rounds. Other baseline algorithms such as FL+Meta_Rl and Krum-only shows slower convergence. Higher residual errors are produced by Vanilla FL. FMRLB has a smooth stable descent below the target error line. This demonstrates faster adaptation with higher final accuracy. These results show the FMRLB's superiority in convergence rate.

Figure 9 shows the privacy preserving capability analysis. The privacy preserving abilities of FMRLB was evaluated and compared with Vainla FL+DP with varying privacy budgets (ε). FMRLB consistently has higher model accuracy with all privacy levels. It reaches the accuracy of 80.5%, at a strict privacy budge of $\varepsilon = 0.1$. When the privacy budget increases, both the method exhibits increased accuracy. Specifically, FMRLB reaches accuracy close to 89% at $\varepsilon = 5.0$, while Vanilla FL+DP remains 81%. These results indicate that FMRLB balances strong privacy with minimal compromise in model accuracy. It tends to be suitable for secure and efficient federated learning deployments.

Figure 10 exhibits the privacy protection capabilities of the methods: FMRLB, FL+MetaRL and Vanilla FL. Renyi divergence ($\alpha = 2$) was used as metric under differential privacy (DP) with $\varepsilon = 0.5$ and without DP. FMRLB exhibits the stronger privacy guarantees having Renyi divergence of 0.12 with DP. Without differential privacy Vanilla FL reaches the highest divergence at 2.5, FL+MetaRL at 2.3 and FMRLB at 2.1. These results highlight the superior ability of FMRLB in limiting privacy leakage.

The proposed algorithm integrates federated learning, meta-reinforcement learning, Byzantine filtering using Krum and LSTM for traffic prediction. Each module introduces a manageable computational overhead. Federated learning incurs a complexity of $O(Bd)$. 'B' is the batch size and 'd' is the model size. Meta-RL adds minimal increase in computation. Since the Krum model needs pairwise distances among models, the complexity of $O(n^2 d)$ per round. 'n' refers to the participant's count. The computational cost of LSTM model is characterized by (Td^2). 'T' is the input size. 'd' is the hidden size. Though, FMRLB has a moderate increase in computational requirement compared to standard FL and RL methods, It has significant gains in convergence speed, robustness against Byzantine attacks, and adaptive traffic handling makes the added cost justifiable.

The FMRLB faces communication overhead from the periodic exchange of model updates during the federated learning process and the aggregation of LSTM parameters for traffic prediction. Each controller transmits locally updated model $\grave{e}_k$ of size d to the aggregator. This results the uplink communication cost of $O(nd)$. Here 'n' denotes the number of participating controllers. There is no additional cost from Byzantine resilient Krum aggregation. Also since meta-learning shares lightweight policy parameters, it introduces negligible additional communication cost. Since LSTM predictor is trained locally, there is no separate communication cost. Hence the communication cost tends to be linear in both number of participants and model size.

Conclusion

This research work titled "Distributed SDN Control for IoT Networks: A Federated Meta-Reinforcement Learning Solution for Load Balancing" is introduced for a distributed IoT-enabled SDN networks. This proposed method incorporates various modules namely, federated learning, meta reinforcement learning, Byzantine resilient aggregation and LSTM traffic prediction. Using these mechanisms the system achieves adaptive, secure and scalable load balancing. The system functions with collaborative policy learning without compromising local data privacy. Byzantine filtering makes the system realisable in adversarial environments. LSTM based traffic helps in forecasting and proactive resource allocation. Theoretical analysis shows that the computational and communicational overhead of the proposed method remains moderate and they are practical for real world SDN environments. FMRLB approach has good convergence speed, robustness and adaptability when comparing other conventional FL and RL based methods. FMRLB serves as a deployable solution for intelligent load balancing in IoT SDN environment.

The FMRLB can be enhanced with dynamic participation of the controllers as the future directions. The controllers may join or leave the federation because of poor network conditions. Federated meta reinforcement can be added with asynchronous update. This further reduces communication delay and it improves responsiveness. Further, block chain assisted trust management can

strengthen the security by providing decentralized and tamper resistant way of verifying model updates.

Acknowledgments

The authors thank, DST-FIST, Government of India for funding towards infrastructure facilities at St. Joseph's College (Autonomous), Tiruchirappalli – 620 002.

References

Albogami, N. N. (2025). Intelligent deep federated learning model for enhancing security in internet of things enabled edge computing environment. *Scientific Reports*, *15*, 4041. https://doi.org/10.1038/s41598-025-88163-5

Allioui, H., & Mourdi, Y. (2023). Exploring the full potentials of IoT for better financial growth and stability: A comprehensive survey. *Sensors (Basel)*, *23*(19), 8015. https://doi.org/10.3390/s23198015

Angelpreethi, A., & Ramesh Kumar, S. B. (2016). Visualizing big datamining: Issues, challenges and opportunities. *International Journal of Control Theory and Applications*, *9*(27), 455–460. https://serialsjournals.com/abstract/86429_61-182.pdf

Bagde, N. K., & Pawar, S. (2024). Software-defined storage performance testing using Mininet. In V. V. S. S. S. Chakravarthy *et al.* (Eds.), *Advances in Microelectronics, Embedded Systems and IoT* (Vol. 1156). Cham, Switzerland: Springer. http://dx.doi.org/10.1007/978-981-97-0767-6_13

Bannour, F., Souihi, S., & Mellouk, A. (2018). Distributed SDN control: Survey, taxonomy, and challenges. *IEEE Communications Surveys & Tutorials*, *20*(1), 333–354. DOI: 10.1109/COMST.2017.2782482

Chien, W.-C., Huang, Y., Chang, B.-Y., & Hwang, W.-Y. (2024). Privacy-Preserving Handover Optimization Using Federated Learning and LSTM Networks. Sensors, 24(20), 6685. https://doi.org/10.3390/s24206685

Huang, V., Chen, G., Zuo, X., Zomaya, A. Y., Sohrabi, N., Tari, Z., & Fu, Q. (2023). Request dispatching over distributed SDN control plane: A multiagent approach. *IEEE Transactions on Cybernetics*, *54*, 3211–3224. https://doi.org/10.1109/tcyb.2023.3266448

Kazmi, M. H., Saad, W., & Bennis, M. (2023). Meta federated reinforcement learning for distributed resource allocation in wireless networks. *arXiv preprint arXiv:2307.02900*. https://arxiv.org/abs/2307.02900

Kazmi, S. H. A., Qamar, F., Hassan, R., Nisar, K., & Chowdhry, B. S. (2023). Survey on joint paradigm of 5G and SDN emerging mobile technologies: Architecture, security, challenges and research directions. *Wireless Personal Communications*, *130*. DOI: 10.21203/rs.3.rs-1648186/v1

Ma, X., Liao, L., Li, Z., Lai, R. X., & Zhang, M. (2022). Applying federated learning in software-defined networks: A survey. *Symmetry*, *14*(2), 195. https://doi.org/10.3390/sym14020195

Madani, S. A., Zia, T. A., Arshad, M. R. K., & Khan, S. (2023). Federated learning augmented cybersecurity for SDN-based networks. *Electronics*, *14*(8), 1–19. https://doi.org/10.3390/electronics14081567

Mahmod, A., Pace, P., & Iera, A. (2025). The role of SDN to improve client selection in federated learning. *IEEE Communications Magazine*, *63*(3), 212–218. https://ieeexplore.ieee.org/document/10697434

Mathanraj, E., Reddy, R. N. (2024). Enhanced principal component gradient round-robin load balancing in cloud computing. *The Scientific Temper*, 15(1):1806-1815. Doi: 10.58414/SCIENTIFICTEMPER.2024.15.1.32

Miguel-Alonso, J. (2023). A research review of OpenFlow for datacenter networking. *IEEE Access, 11*, 770–786. https://doi.org/10.1109/ACCESS.2022.3233466

Rahim, J. A., Nordin, R., & Amodu, O. A. (2024). Open-source software defined networking controllers: State-of-the-art, challenges and solutions for future network providers. *Computers, Materials & Continua*, *80*(1), 747–800. https://doi.org/10.32604/cmc.2024.047009

Raza, M., Saeed, J., Riaz, M., & Sattar, M. (2024). Federated learning for privacy preserving intrusion detection in software defined networks. *IEEE Access*, *12*, 69551–69567. Doi: 10.1109/ACCESS.2024.3395997

Sinduja, V.I., Charles, P.J. (2025). A hybrid approach using attention bidirectional gated recurrent unit and weight-adaptive sparrow search optimization for cloud load balancing. *The Scientific Temper*, 16(5):4270-4283.

Tran, H. A., & Tran, D. (2024). FL4SDN: A fast-convergent federated learning for distributed and heterogeneous SDN. *IEEE Intelligent Systems*, 1–28. http://dx.doi.org/10.1109/MIS.2024.3453308

Wang, Y., Zhao, S., & Hu, H. (2024). WFAgg: A Byzantine-robust aggregation algorithm for decentralized federated learning. *arXiv preprint arXiv:2409.17754*. https://arxiv.org/abs/2409.17754

Zhou, X., Yang, J., Li, Y., Li, S., & Su, Z. (2024). Deep reinforcement learning-based resource scheduling for energy optimization and load balancing in SDN-driven edge computing. *Computer Communications, 226–227*, 107925. https://doi.org/10.1016/j.comcom.2024.107925

Zhuang, Z., Lei, K., Liu, J., Wang, D., & Guo, Y. (2023). Behavior proximal policy optimization. In *International Conference on Learning Representations(ICLR)*. https://doi.org/10.48550/arXiv.2302.11312

Zormati, M. A., & Lakhlef, H. (2023). Enabling architecture for distributed intelligent network softwarization for the internet of things. In *Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)* (pp. 608–609). http://dx.doi.org/10.1109/MASS58611.2023.00082