Network Traffic Prediction in Edge-Cloud Continuum Network for Multiple Network Service Providers

Ying Hu *, Ben Liu, Jianyong Li, Liang Zhu, Jihui Han, Zengyu Cai, Jie Zhang

Posted Date: 18 July 2024

doi: 10.20944/preprints202407.1484.v1

Keywords: Edge-Cloud Continuum; Network Function Virtualization; Virtual Network Function Migration; Federated Learning

Article

Network Traffic Prediction in Edge-Cloud Continuum Network for Multiple Network Service Providers

Ying Hu *, Ben Liu, Jianyong Li, Liang Zhu, Jihui Han, Zengyu Cai and Jie Zhang

School of Computer Science and Technology, Zhengzhou University of Light Industry, Zhengzhou 450001, China; ccehuying@zzuli.edu.cn (Y.H.); 332305090541@email.zzuli.edu.cn (B.L.); lijianyong@zzuli.edu.cn (J.L.);
* Correspondence: ccehuying@zzuli.edu.cn

Abstract: Network Function Virtualization (NFV) allows for the dynamic provisioning of Virtualized Network Functions, adapting services to the complex and real-time network environment for enhanced network performance. Despite the benefits of NFV, virtual network functions (VNFs) migration and the energy consumption pose significant challenges due to the dynamic nature of the physical network, especially in the edge-cloud continuum network. Currently, people tackle this challenge by predicting the network traffic and then proactively migrating the VNFs using the predicted values. There are two challenges here. First, when SFCs belong to different network service providers, the historical network traffic data is held by the service providers, and they are reluctant to share the historical data due to privacy concerns. Without the historical network traffic data of each SFC, the resource provider that owns the underlying network cannot effectively predict the network traffic; in addition, without the feedback of the migration results of the underlying physical network, the network traffic prediction will be difficult to match the needs of the physical network. In this manuscript, we solve the problem of network traffic prediction in the NFV-based edge-cloud continuum network for multiple network service providers. We aim to improve migration performance and reduce energy consumption while ensuring resource and latency requirements. To protect the privacy of each network service provider, we apply an FL algorithm. After that, to improve the migration performance, we tailor the loss function. In the loss function, factors such as the number of migrations, the number of migration failures, and the energy consumption of the NFV-based edge-cloud continuum network are added. Additionally, to obtain the results of the SFC migration, we develop a mathematical model for the problem of multiple network service providers in the NFV-based network. The effectiveness of our algorithm is evaluated through extensive simulations, and the results show a significant reduction in the number of migrated nodes and energy consumption while improving the SFC acceptance ratio compared to state-of-the-art schemes which only use the difference between predicted and actual flows to define the loss function.

Keywords: edge-cloud continuum; network function virtualization; virtual network functions migration; federated learning

1. Introduction

Network Function Virtualization (NFV) is a revolutionary technology that leverages cloud computing and virtualization techniques to decouple traditional network hardware and software [1]. This decoupling allows for the rapid development and deployment of new network services by telecom operators in a more cost-effective and flexible manner [2]. By virtualizing network functions and running them as Virtual Network Functions (VNFs) on commodity hardware [3], NFV significantly enhances network agility, scalability, and cost efficiency. The edge-cloud continuum network [4,5], enabled by NFV, can cater to a wide range of applications with varying performance and latency requirements. From industrial automation to smart cities, the network's adaptability and scalability ensure optimal performance across diverse use cases. The edge-cloud continuum network based on NFV represents a significant advancement in network architecture. By leveraging the benefits of both edge computing and NFV, it enables the delivery of high-performance, cost-effective, and flexible network services. The integration of these technologies holds immense potential for driving innovation, enhancing user

 doi:10.20944/preprints202407.1484.v1

experiences, and supporting the diverse needs of modern digital ecosystems [6]. Network traffic in the edge-cloud continuum network is dynamically changing, and the changing network traffic will lead to passive migration of SFCs. In order to reduce passive migration, a day can be divided into multiple time slices, network traffic is predicted for each time slice at the beginning of the time slice, and network resources are reallocated using the predicted value. If the predicted value of network traffic for each time slice is greater than the actual network traffic value, it will occupy unneeded resources, resulting in a waste of resources and increasing network energy consumption; if the predicted value of network traffic is less than the actual network traffic value, it can not avoid passive migration within the time slice. Thus, network traffic prediction in the NFV environment is one of the key issues to determine the performance of the network. In addition, the network service providers to which each SFC belongs are usually different from each other, and the network resource providers that provide resource support to SFCs are usually different from the network service providers. Due to privacy concerns, each network service provider is reluctant to share their historical network traffic data. Thus, to enable network resource providers to better predict network traffic, we apply a Federated Learning (FL) algorithm to enable each network service provider to locally train a neural network model and upload the trained local model to the network resource service provider, which aggregates the received multiple local models to obtain a global model and uses the global model to provide network traffic prediction in edge-cloud continuum network. In this paper, we apply FL to predict network traffic for different network service providers in the edge-cloud continuum network. The main contributions of this paper are summarized as follows:

(1) In order to protect the privacy of historical network traffic data for different network service providers, we propose a FL-based network traffic prediction algorithm (Newloss_FL_NTP) to train the neural network model for the network resource provider. In the Newloss_FL_NTP algorithm, the loss function is redesigned to include the results of network migration in calculating of the loss function, which allows the migration effect of the network to be fed back into the training of the network traffic prediction. In this paper, we apply FL to predict network traffic for different network service providers in edge-cloud continuum network.

(2) In order to feed the migration results to the neural network model used for network traffic prediction, we redesigned the loss function. This loss function adds factors such as the number of migrated nodes, the number of failed migration requests, and energy consumption to make the network traffic prediction models' strategy more efficient in terms of migration.

(3) In order to achieve migration awareness, We formulate the SFC deployment (SFCD) and SFC migration (SFCM) problems for multiple network service providers, which consider resource availability, end-to-end latency, energy consumption and migration in edge-cloud continuum network.

(4) We propose greedy mapping and migrating algorithms for training of the network traffic prediction model to adapt to dynamic network traffic in the edge-cloud continuum network.

(5) Finally, we perform extensive simulations using a simulated network topology of the edge-cloud continuum to evaluate the proposed algorithms. Simulation results demonstrate that our algorithms can reduce the number of migrated nodes and the energy expenditure compared with the state of the art mechanisms.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work. In Section 3 we give the formulation of the SFCD and SFCM problems. Section 4 demonstrates our framework and algorithms designed for predicting network traffic in the edge-cloud continuum network. Additionally, Section 5 is the performance evaluation of the algorithms. Finally, we conclude the paper in Section 6.

2. Related Works

In NFV-based networks, there have been many researches on the problem of dynamic network traffic. Work [7] proposed a neural network-based model with adaptive spatial-temporal analysis to predict the future traffic in NFV networks. Work [8] explored Machine Learning approaches that can

 doi:10.20944/preprints202407.1484.v1

be used to predict the traffic of a cloud platform that uses an SDN architecture and offers VNFs as services, and they focused on Facebook (FB) prophet approach and Random Forest (RF) regression models. In work [9], the traffic load forecast is achieved by using and training a neural network on a real dataset of traffic arrival in a mobile network, and two techniques were used and compared: LSTM and DNN. Work [10] trained models on actual VNF data, which can accurately predict the number of resources needed for each VNF to handle a specific traffic load. Work [11] proposed and investigated a forecasting methodology in which it is introduced an asymmetric cost function capable of weighing the costs of over-provisioning and under-provisioning differently. Work [12] proposed VNF requirement prediction methods based on Deep Neural Networks (DNN) and Long Short Term Memory (LSTM) Networks. Work [13] proposed a negotiation-game-based auto-scaling method where tenants and network operators both engage in the auto-scaling decision, based on their willingness to participate, heterogeneous QoS requirements, and financial gain (e.g., cost savings). In addition, they proposed a proactive Machine Learning (ML) based prediction method to perform SC auto-scaling in dynamic traffic scenario. Work [14] proposed and evaluated a traffic forecasting based dynamic scaling scheme using ML. The algorithms aimed at predicting future O-RAN traffic by using previous traffic data, then utilized the prediction in coming up with a scaling decision. Work [15] outlined a vision for leveraging FL for better traffic steering predictions. Specifically, they proposed a hierarchical FL framework that will dynamically update service function chains in a network by predicting future user demand and network state using the FL method. Work [16] formulated the VNF provisioning problem in order that the cost incurred by inaccurate prediction and VNF deployment is minimized. Work [17] proposed the model for an Asynchronous Deep Reinforcement Learning (DRL) enhanced Graph Neural Networks (GNN) for topology-aware VNF resource prediction in dynamic NFV environments. Work [18] proposed a traffic forecasting model using LSTM networks and used it to place VNFs accordingly to the predicted traffic demands, then they proposed an offline MILP model as well as an online greedy algorithm for the placement optimization problem. Work [19] proposed a novel traffic model for cloud-oriented transport networks, and proposed a dedicated allocation algorithm. Work [20] presented a cloud-native architecture, with traffic prediction capabilities through dynamic analysis of real-time deployed connectivity services, and the proposed architecture has been tested using four completely different ML algorithms.

FL algorithms have been applied to many problems in edge-cloud continuum networks [21,22]. Work [21] investigated the FL-based Smart Decision Making module for ECG Data in microservice-based IoT medical applications. In addition, they also examined the performance of the proposed system with three different placement policies considering the deployment at Edge, Fog and Cloud layers. Work [22] introduced the kubeFlower operator for FL in cloud-edge environments, which features isolation-by-design, differentially private data consumption and simplifies the life-cycle management of FL applications within Kubernetes clusters, allowing automated scaling and efficient configuration.

 doi:10.20944/preprints202407.1484.v1

Table 1. Comparison of related work.

Paper	Network traffic or resource prediction	NFV-based network	Environment	Technique	Interact with the environment	Metrics						Multiple network service provider
						The number of migrated nodes	Acceptance ratio	Energy consumption	Cost	End-to-end latency	Other metrics	
[7]	✓	✓		LSTM								
[8]	✓	✓		FBFR								
[9]	✓	✓		LSTMDNN								
[10]	✓	✓		ML								
[11]	✓	✓		LSTM	✓			✓				
[12]	✓	✓		LSTMDNN								
[13]	✓	✓		Negotiation-game	✓			✓				
[14]	✓	✓	5G mobile communication	ML								
[15]	✓	✓	Data Centers	FL								
[16]	✓	✓	Cloud computing	An online heuristic algorithm	✓			✓				
[17]	✓	✓		DRLGNN							Topology	
[18]	✓	✓	Cloud computing	LSTM								
[19]	✓		Intent-based elastic optical networks	Supervised learning								
[20]		Transport networks		ML								
Our proposal	✓	✓	✓	FLLSTM	✓	✓	✓	✓	✓	✓		✓

Existing research forms the basis of our study and provides inspiration for solving the dynamic network traffic problem. Our research focuses on the problem of network traffic prediction in NFV-based edge-cloud continuum network under dynamic network traffic while considering the protection of historical network traffic data from different network service providers. In this manuscript, we train local models for different network service providers using their own historical network traffic data, targeting resource and latency guarantees, and optimizing energy and migration in the edge-cloud continuum network. Considering dynamic changes in network traffic, we pursue provable algorithms for network traffic prediction in the edge-cloud continuum network. To the best of our knowledge, this is the first work to address this topic in the NFV-based edge-cloud continuum network.

3. Problem Formulation

3.1. Use Cases

A variety of applications including smart transportation, smart healthcare, smart government, industrial internet, and so on, can be realized in the Edge-Cloud Continuum Network. Here, we select the first two applications for illustration as follows.

In the field of smart healthcare, edge devices can monitor patients' physiological data in real time, and transmit the data to medical experts for remote diagnosis through cloud-edge continuum network, which can provide doctors with more efficient diagnosis and treatment services. In the field of smart transportation, by completing some or all of the computing tasks at the edge closer to the camera, the video analysis system can complete the video analysis tasks with lower bandwidth consumption and lower latency. The cloud computing center, on the other hand, is responsible for collecting data from the widely distributed edge nodes, sensing the operating conditions of each system, and issuing reasonable scheduling instructions for the traffic signal system through algorithms such as artificial intelligence, thereby improving the operational efficiency of the system. Cloud servers for smart healthcare act as medical experts to provide diagnosis and health management solutions for remote edge devices; cloud servers for smart cities have data from widely distributed edge nodes and issue scheduling instructions for end nodes such as traffic signals.

In the field of smart healthcare applications, smart healthcare service requests usually occur at the time of scheduling consultations or surgeries in hospitals, and the time of use of surgeries or consultations is usually realized arranged and relatively concentrated, so the network traffic of such applications is subject to a certain pattern of change. In the field of smart transportation applications, the daily traffic flow at each signal light is also highly regular and cyclical. Whether in the field of smart healthcare applications or intelligent transportation applications, flows are somewhat periodical

 doi:10.20944/preprints202407.1484.v1

and regular. Therefore, it is necessary for network services to plan and allocate network resources according to their network traffic.

As mentioned above, on the one hand, the network traffic of many network services is somewhat periodic and regular; on the other hand, the training of neural network models is placed on cloud servers, and the training and prediction of the models can be done in parallel. Therefore, the algorithm proposed in this paper can be applied to realize network traffic prediction for certain network services. Additionally, in the above two applications, the cloud servers provide different types of services, and therefore they possibly belong to different network service providers. Network function virtualization technology enables multiple service function chains from different service providers to share the same physical network resources. Since the demand for individual network services changes over time, in order to deploy service function chain requests more accurately, network resource providers need to know the network traffic of service function chains from different service providers. However, different network service providers are reluctant to share their network traffic data due to privacy concerns, so we propose applying FL algorithms to solve this problem.

3.2. System Model

Table 2 summarizes the notations used in this paper.

Table 2. The notations.

M	Number of edge servers
Q	Number of cloud servers
V_i	The ith edge server
V'_s	the sth cloud server
$E_{i,j}$	The link between edge server V_i and edge server V_j
$E'_{s,i}$	The link between edge server V_i and cloud server V'_s
C_i	The processing capacity of edge server V_i
C'_s	The processing capacity of cloud server V'_s
$B_{i,j}$	The bandwidth capacity of link from edge server V_i to edge server V_j
$B'_{s,i}$	The bandwidth capacity of link between edge server V_i and cloud V'_s
$d_{i,j}$	Communication latency for transmitting data from edge server V_i to edge server V_j
$d'_{s,i}$	Communication latency for transmitting data between edge server V_i and cloud V'_s
m	Number of SFCs
Q'	number of network service providers
$n_{s',i}$	Number of VNFs in SFC i which belongs to network service provider s'
$F_{s',i,j}$	VNF j in SFC i of network service provider s'
$c_{s',i,j}$	The requirement of computing resource for VNF $F_{s',i,j}$
$d^k_{s',i,j}$	The processing time of VNF $F_{s',i,j}$ on edge server V_k or cloud server V'_k
$b_{s',i,j}$	The bandwidth requirement of data flow between $F_{s',i,j}$ and $F_{s',i,j+1}$
$d_{s',i,j}$	The latency of data flow between $F_{s',i,j}$ and $F_{s',i,j+1}$
$x^k_{s',i,j}$	If VNF $F_{s',i,j}$ is placed on V_k or V'_k or not
y_k	if server V_k or V'_k is occupied or not
f_k	if server V_k or V'_k is a cloud server or not
$z^{p,q}_{s',i,j}$	If data flow between VNF $F_{s',i,j}$ and $F_{s',i,j+1}$ pass through link $E_{p,q}$ (or $E'_{p,q}$) or not
$f_{p,q}$	If V_p or V_q is a cloud server or not
$b_{p,q}$	The total amount of occupied bandwidth of link $E_{p,q}$ or $E'_{p,q}$
$D'_{s',i}$	The latency limit of SFC i which belongs to network service provider s'
$D_{s',i}$	The actual latency of SFC i which belongs to network service provider s'
$d^{p,q}_{s',i,j}$	The latency of data flow from $F_{s',i,j}$ to $F_{s',i,j+1}$ on physical link $E_{p,q}$ or $E'_{p,q}$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 July 2024 doi:10.20944/preprints202407.1484.v1

Table 2 *Cont.*

V^s	Starting node of SFC
V^d	Destination node of SFC
$k_{s',i,j-1}$	Resident server for $F_{s',i,j}$
S	The number of time slices in one day

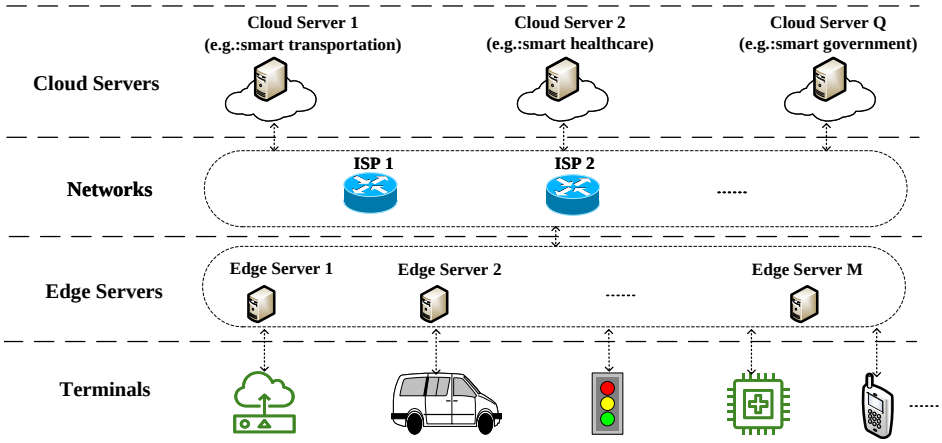


Figure 1. Topology.

We consider an edge-cloud continuum network represented as a connected graph $G = ((V + V'), E)$. Figure 1 illustrates the topology of the edge-cloud continuum network. In graph G, the vertex V_i denote the *ith* commercial edge server , and the link $E_{i,j}$ denote there is a directly connection between edge server V_i and edge server V_j in the network. The vertex V'_s denote the commercial cloud server s, and the link $E'_{s,i}$ denote there is a directly connection between edge server V_i and cloud server V'_s in the network. We assume that there are M servers located at the edge, and Q servers located at the cloud. The processing capacity of edge server V_i is C_i, and the processing capacity of cloud server V'_s is C'_s. If edge servers V_i and V_j are directly connected, a physical link $E_{i,j}$ exists in the graph between the two vertices, whose bandwidth capacity is $B_{i,j}$, and the communication latency for transmitting data from V_i to V_j is $d_{i,j}$. We assume the cloud server V'_s is directly connected to the target edge server V_i. If edge server V_i and cloud server V'_s are directly connected, a physical link $E'_{s,i}$ exists in the graph, whose bandwidth capacity is $B'_{s,i}$, and the communication latency for transmitting data from V_i to V_j is $d'_{s,i}$.

There are multiple SFCs requested by users, and these SFCs belong to different network service providers. Each SFC comprises an ordered set of VNFs. We assume there are m SFCs, and they belong to Q' network service providers. Each SFC i comes from network service provider s' contains $n_{s',i}$ VNFs, and the VNFs in the SFC i are noted as $F_{s',i,1}$, $F_{s',i,2}$,...,$F_{s',i,n_{s',i}}$ in intended order. The requirement of computing resource for VNF $F_{s',i,j}$ is $c_{s',i,j}$, which is the same on edge and cloud. The processing time of VNF $F_{s',i,j}$ is different in edge and cloud server (the server is k), which is denoted as $d^k_{s',i,j}$. The processing time of the VNF on edge nodes is longer than the processing time on the cloud [23]. In each SFC, there is data flow between two neighboring VNF. There is a data flow between $F_{s',i,j}$ and $F_{s',i,j+1}$ in each SFC. A network traffic flow occupies bandwidth while incurring transmission delay. Where the bandwidth consumed by the network traffic is $b_{s',i,j}$, and the transmission delay incurred is $d_{s',i,j}$. When some VNFs are offloaded to the cloud, communication latency between the edge and cloud arises, referred to as E&C communication latency [23]. Most servers cannot "directly" communicate with the cloud, such communication latency typically has two parts, which include the inside-edge part and the outside-edge part [23]. We assume each cloud server directly connected to an edge server, and the cloud server of each network service provider use different edge server to connect to edge network, and for any cloud server there is no "directly" connection except the connection from the target server in edge network to its cloud server [23]. The inside-edge part for cloud server V'_s is from

 doi:10.20944/preprints202407.1484.v1

start edge server V_j to edge target server V_i , and the outside-edge part is the direct link from the edge target server V_i to the cloud server V'_s.

3.3. Problem Definition

1) Service Function Chain Deployment

The Service Function Chain Deployment (SFCD) problem is to place the m SFCs to the edge servers or cloud server without exceeding the constraints of resource capacities and the latency limits of the physical network. Our goal of SFCD problem is to minimize energy consumption and guarantee the inviolability of resource and network latency constraints. In particular, the total network latency of a SFC includes the total processing latency, the communication latency between VNFs placed at the edge or that between the edge and cloud. The total energy consumption contains the total energy consumed on the occupied edge servers and the offloaded clouds.

The capacity constraint of each server asks

$$\sum_{i=1}^{m}\sum_{j=1}^{n_{s',i}}(1-f_k)\cdot x^k_{s',i,j}\cdot c_{s',i,j}+\sum_{i=1}^{m}\sum_{j=1}^{n_{s',i}} f_k\cdot x^k_{s',i,j}\cdot c_{s',i,j}\le(1-f_k)\cdot y_k\cdot C_k+f_k\cdot y_k\cdot C'_k,\quad \forall 1\le k\le\max(M,Q) \tag{1}$$

where $x^k_{s',i,j}=1$ if and only if VNF $F_{s',i,j}$ is placed on edge server V_k or cloud V'_k; $y_k=1$ if and only if server V_k or V'_k is occupied; $f_k=1$ if and only if server V_k or cloud V'_k is a cloud server.

The capacity constraint of each link requires

$$b_{p,q}=\sum_{i=1}^{m}\sum_{j=0}^{n_{s',i}}(1-f_{p,q})\cdot z^{p,q}_{s',i,j}\cdot b_{s',i,j}+\sum_{i=1}^{m}\sum_{j=0}^{n_{s',i}} f_{p,q}\cdot z^{p,q}_{s',i,j}\cdot b_{s',i,j}\le(1-f_{p,q})\cdot B_{p,q}+f_{p,q}\cdot B'_{p,q},\quad \forall 1\le p,q\le\max(M,Q) \tag{2}$$

where $z^{p,q}_{s',i,j}=1$ if and only if data flow between VNF $F_{s',i,j}$ and $F_{s',i,j+1}$ pass through link $E_{p,q}$ (or $E'_{p,q}$) ; $f_{p,q}=1$ if and only if V_p or V_q is a cloud server; $b_{p,q}$ is the occupied bandwidth of link $E_{p,q}$ (or $E'_{p,q}$).

Each VNF is exactly placed on an edge server or the cloud, that is, it can not be split, we obtain

$$\sum_{k=1}^{Q} f_k\cdot x^k_{s',i,j}+\sum_{k=1}^{M}(1-f_k)\cdot x^k_{s',i,j}=1,\quad \forall 1\le i\le m, 1\le j\le n_{s',i} \tag{3}$$

It must follow the Flow Conservation Law for the data flow between VNF $F_{s',i,j}$ and $F_{s',i,j+1}$, we obtain

$$\begin{aligned}&(1-f_p)\cdot(1-f_q)\cdot\left(\sum_{p=1}^{M} z^{p,q}_{s',i,j}-\sum_{q=1}^{M} z^{q,p}_{s',i,j}\right)\\&+(1-f_p)\cdot f_q\cdot\left(\sum_{p=1}^{M} z^{p,q}_{s',i,j}-\sum_{q=1}^{Q} z^{q,p}_{s',i,j}\right)\\&+f_p\cdot(1-f_q)\cdot\left(\sum_{p=1}^{Q} z^{p,q}_{s',i,j}-\sum_{q=1}^{M} z^{q,p}_{s',i,j}\right)\\&=x^q_{s',i,j+1}-x^q_{s',i,j}\end{aligned} \tag{4}$$

The total latency includes the communication latency and the processing latency, and it can not exceeds the latency limit. Therefore, we obtain

$$D_{s',i}=\sum_{j=0}^{n_{s',i}} d_{s',i,j}+\sum_{j=1}^{n_{s',i}}\sum_{k=1}^{\max(M,Q)}\left[(1-f_k)\cdot\left(d^k_{s',i,j}\cdot x^k_{s',i,j}\right)+f_k\cdot\left(d^k_{s',i,j}\cdot x^k_{s',i,j}\right)\right]\le D'_{s',i},\quad \forall 1\le i\le m \tag{5}$$

 doi:10.20944/preprints202407.1484.v1

where $D'_{s,i}$ denotes the latency limit in SFC i of network service provider s'. The communication latency $d_{s',i,j}$ that data flow passes through from $F_{s',i,j}$ to $F_{s',i,j+1}$ is

$$d_{s',i,j} = \sum_{p=1}^{\max(M,Q)} \sum_{q=1}^{\max(M,Q)} \left(f_p \cdot (1-f_q) \cdot z^{p,q}_{s',i,j} \cdot d^{p,q}_{s',i,j} + (1-f_p) \cdot f_q \cdot z^{p,q}_{s',i,j} \cdot d^{p,q}_{s',i,j} + (1-f_p) \cdot (1-f_q) \cdot z^{p,q}_{s',i,j} \cdot d^{p,q}_{s',i,j} \right), \quad \forall 1 \le i \le m, 0 \le j \le n_{s',i} \tag{6}$$

where $d^{p,q}_{s',i,j}$ denotes the latency of data flow from $F_{s',i,j}$ to $F_{s',i,j+1}$ on physical link $E_{p,q}$ or $E'_{p,q}$. If there is edge and cloud communication latency between $F_{s',i,j}$ and $F_{s',i,j+1}$, the data flow determine the transmission path in two steps as described in the previous section. In addition, if the first VNF of a SFC is offloaded to the cloud, there exists a transmitting data flow from the initial edge server to the VNF on the cloud. Similarly, if the last VNF of a SFC is offloaded to the cloud, there is a transmitting data flow from the last VNF on the cloud to the final edge.

In our SFCD model, the optimization objective is to minimize the energy cost in the network. The energy on the edge server is

$$E_k = e_1 + (e_1 - e_0) \cdot \left((1-f_k) \cdot \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_{s',i}} x^k_{s',i,j} \cdot c_{s',i,j}}{C_k} + f_k \cdot \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_{s',i}} x^k_{s',i,j} \cdot c_{s',i,j}}{C'_k} \right) \tag{7}$$

where e_0 denotes minimum energy consumption when the node is turned on, e_1 denotes maximum energy consumption when the node is turned on.

The energy on the network is approximately defined as

$$E = \sum_{k=1}^{Q} f_k \cdot E_k + \sum_{k=1}^{M} (1-f_k) \cdot E_k \tag{8}$$

In all, the SFCD problem can be formulated as the below Integer Linear Programming (ILP) problem.

$$\begin{aligned} &\min \quad E \\ &s.t. (1) - (5) \end{aligned} \tag{9}$$

2) Service Function Chain Migration

The Service Function Chain Migration(SFCM) problem is to migrate the m SFCs according to the dynamic data flow which generate dynamic network resource requirements and dynamic communication latency. The goal of SFCM problem is to minimize migrated nodes, energy consumption and network latency. In particular, the total energy consumption contains the total energy consumed on the occupied edge servers or cloud servers. The total number of migrated nodes contain the migrated nodes compared with the network before the migration. That is, in our model, there are below three optimization objectives.

Network energy E defined in Equation (8),

The total number of migration nodes M defined in Equation (10),

$$M = \sum_{i=1}^{m} \sum_{j=1}^{n_{s',i}} \left(x^k_{s',i,j} - x^{k'}_{s',i,j} \right) \cdot x^k_{s',i,j} \cdot x^{k'}_{s',i,j} \tag{10}$$

The total number of failed SFC requests F defined in Equation (11),

$$F = \sum_{i=1}^{m} (1 - S_{s',i}) \tag{11}$$

 doi:10.20944/preprints202407.1484.v1

where k' is the placed node before the migration; $S_{s',i} = 1$ if and only if the SFC i of network service provider s' is successfully placed.

In all, the SFCM problem can be formulated as the below Integer Linear Programming (ILP) problem.

$$\begin{aligned} \min \quad & \ell = \alpha E + \beta M + \gamma F \\ s.t. & (1) - (5) \end{aligned} \tag{12}$$

Where α, β, γ are weighting factors for adjusting the relative importance between objective components.

4. Framework and Algorithms

This section presents in detail the proposed framework and algorithms for network traffic prediction in edge-cloud continuum network under dynamic network traffic. First, we present the framework of the solution for dynamic network traffic in edge-cloud continuum network and then introduce the algorithms of the solution in detail.

4.1. Framework for Traffic Prediction in Edge-Cloud Continuum Network

In NFV-based edge-cloud continuum network, the network resource requirements of SFCs and the transmission delay of links change according to dynamic network traffic. In order to solve this problem, we propose to divide a day into S time slices, predict the network traffic of the next time slice when the time slices are alternating, get the network resource requirements and possible transmission delays of the next time slice based on the predicted network traffic, and proactively migrate the VNFs to comply with the traffic changes of the next time slice, so as to reduce the amount of passive migration within the time slice and reduce the growth of energy consumption.

The main idea of the scheme is to design an intelligent framework under dynamic network traffic with the goal of protecting privates of network service providers. By utilizing FL to train multiple local network traffic prediction models belonging to different network service providers, the local models can implement optimal network traffic prediction using history network traffic.

The framework of our scheme for network traffic prediction is illustrated in Figure 2. There are two kinds of servers, i.e., the cloud servers and the edge servers. Among them, the cloud servers can be used as a supplement to the edge servers, as a carrying host for VNFs when the edge servers do not have enough resources for SFC requests; in addition, the cloud servers can be used as a host for the aggregation models in federated learning, when the edge servers are in charge of local models for each network service provider, thus supporting the federated learning algorithms for network traffic prediction.That is, the cloud servers are responsible for global model training and VNFs hosting, while the edge servers are in charge of local model training and VNFs hosting. The cloud servers are responsible for aggregation global training and VNFs sustaining, while the edge servers are in charge of local training and VNFs sustaining. Each day is made of series of episodes $T = \{t_1, t_2, ...\}$, each of which has a time slot, and the whole day has been divided into S time slots. There are Q' network service provider, each of which has a local server to iteratively learn its local model $\theta_i (i \in [1, Q'])$ via FL and serves as a function node to communicate with the cloud server. The model θ_i is structured based on neural network of LSTM, guiding network traffic prediction for each time slot. The network traffic prediction and SFC migration aims to actively migrate VNFs with the dynamic network traffic of different time slots. The initial global model will be delivered to local cloud server from the cloud server. The local training executed by each local cloud server is to learn a locally traffic prediction model θ_i using LSTM. To preserve the privacy of network traffic data, each local model θ_i is not share history data with other models. The aggregation training is to collect all local models θ_1 ,..., $\theta_{Q'}$ executed by the local cloud server for the global model θ_0 learning and then send the global model to local models for further network traffic prediction. Essentially, θ_0 has an LSTM-based architecture of the global model with a number of parameters learned from all local models. Through the collaborative efforts among them, it can train the global model to predict network traffic.

 doi:10.20944/preprints202407.1484.v1

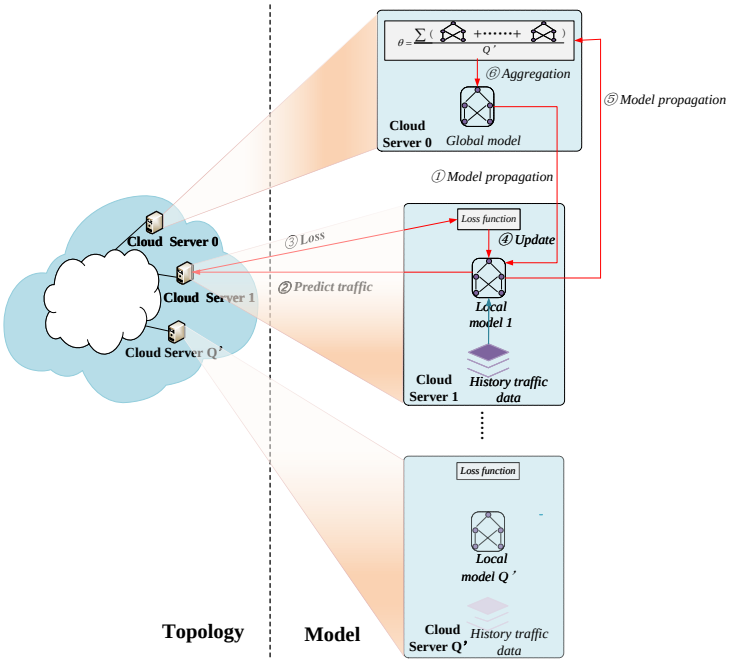


Figure 2. The Framework.

The operation of the network traffic prediction is generally divided into six steps: ① global model propagation, ② network traffic prediction, ③ loss computation, ④ update local model, ⑤ local model propagation, ⑥ model aggregation.

In step ①, the global model is delivered to different local models. In step ②, Each model mainly consists of LSTM layers with local model parameters, guiding the local server to learn a model θ_i and predict network traffic as the time slot changes. In step ③, the local model output the predicted network traffic into the network, and migrate VNFs according to the changed requirements caused by dynamic network traffic. Then, it compute the loss according to the migration results. In step ④, input the loss into the local model, then compute and update the local model θ_i . In step ⑤, the parameters of each local model are delivered to the global model. In step ⑥, the global model aggregates the updated parameters of local models received from edge servers into a new model θ_0 according to Equation (13). The above steps are repeated to promote the evolution of the models.

In our FL-based framework, the update method of global model θ_0 can be expressed as

$$\theta_0 = \frac{\sum_{i=1}^{Q'} \theta_i}{Q'} \tag{13}$$

4.2. Algorithms for Network Traffic Prediction in Edge-Cloud Continuum Network

There are two main problems involved in prediction of network traffic, including the model training for prediction, and the computation of the loss function. In order to get more accurate loss values, the loss function calculation can rely on the actual migration results of the network. The calculation of loss function mainly involves the placement of SFC and the migration of SFC, and the problems have been formulated in Section 3.3. We describe the three implementation processes of SFC placement, SFC migration and prediction model training in detail as follows.

4.2.1. First Fit Algorithm of SFCD

We can rely on the powerful computing resource of the cloud to reduce processing latency for some special SFCs. Otherwise, we prioritize edge resource due to their low communication latency. To maximize the exploitation of edge resources and reduce the complexity of the algorithm, we have the

 doi:10.20944/preprints202407.1484.v1

same idea with [24,25] for SFCD, which is to prioritize VNFs at the edge as resources allow and then offload all the remaining VNFs onto the cloud. The detailed procedures of the First Fit SFCD algorithm are shown in Algorithm 1.

Algorithm 1 First Fit (FF) Algorithm of SFCD in edge-cloud continuum network

Input: G, the edge servers V_1, V_2 ,...,V_M , the cloud servers V'_1 , V'_2 ,..., V'_Q , the list of VNFs $\left\{F_{s',i,j}\right\}$ of the SFC, the capacities of the network.
Output: The placements $\left\{x^k_{s',i,j}\right\}$ and $\left\{z^{p,q}_{s',i,j}\right\}$.
for each VNF $F_{s',i,j}$ in $n_{s',i}$ **do**
  the sustaining server $k_{s',i,j-1}$ of VNF $F_{s',i,j-1}$ has the highest priority;
  if $f_{k_{s',i,j-1}} = 0$ ($k_{s',i,j-1}$ is an edge server) **then**
    add all available edge servers to the list l and sort them by resource utilization in descending order;
  else
    the only node that enters the list l is $k_{s',i,j-1}$;
  end if
  for each server k in list l: **do**
    map to server k,
    if it is successful **then**
      go to step 15;
    else
      go to step 29;
    end if
    map the link between VNF $F_{s',i,j}$ and VNF $F_{s',i,j-1}$, and obtain the physical path consisting of a sequence of physical links $\left\{z^{p,q}_{s',i,j}\right\}$,
    if it is successful **then**
      go to step 21;
    else
      go to step 29;
    end if
    if $j = n_{s',i}$ **then**
      map the link between VNF $F_{s',i,j}$ and the destination node V^d $\left\{z^{p,q}_{s,i,n_{s,i}+1}\right\}$,
      if it is successful **then**
        go to step 32;
      else
        go to step 29;
      end if
    end if
    Clear the mapping of VNF $F_{s',i,j}$ on the server;
    if the link between VNF $F_{s',i,j}$ and VNF $F_{s',i,j-1}$ is successfully mapped **then**
      clear the mapping;
      if both nodes and links are successfully mapped **then**
        break;
      else
        continue;
      end if
    end if
  end for
end for
if all of the VNFs and virtual links are successfully mapped **then**
  Fail=False;
else
  Release resources occupied by mapped VNFs and VLs in the physical network;
  Fail=True;
end if
return Fail and the placements $\left\{x^k_{s',i,j}\right\}$ and $\left\{z^{p,q}_{s',i,j}\right\}$

4.2.2. First Fit Algorithm of SFCM

In order to reduce the complexity of the algorithm, we use the greedy algorithm to solve the problem of SFCM. To reduce the number of migrated nodes, the previously mapped nodes are set as the highest priority level. At the same time, to reduce the energy consumption, the one with a large percentage of resource consumption is set as the next highest priority. The detailed procedure for SFCM is shown in Algorithm 2.

 doi:10.20944/preprints202407.1484.v1

Algorithm 2 First Fit (FF) Algorithm of SFCM in edge-cloud continuum network

Input: G, the edge servers V_1 , V_2 ,..., V_M , the clouds V'_1 , V'_2 ,..., V'_Q , the list of VNFs $\left\{F_{s',i,j}\right\}$, the capacities of the network $c_{s',i,j}$ and $b_{s',i,j}$, the processing latency $d^k_{s',i,j}$ and the transmitting latency $d_{s',i,j}$, the placements $\left\{x^k_{s',i,j}\right\}$ and $\left\{z^{p,q}_{s',i,j}\right\}$, the predicted network traffic p_t and the network traffic p_{t-1} for the previous time slice.
Output: The placements $\left\{x^k_{s',i,j}\right\}$ and $\left\{z^{p,q}_{s',i,j}\right\}$ after migration .

$c_{s',i,j} \leftarrow c_{s',i,j} \cdot \frac{p_t}{p_{t-1}}$, $b_{s',i,j} \leftarrow b_{s',i,j} \cdot \frac{p_t}{p_{t-1}}$, $d_{s',i,j} \leftarrow d_{s',i,j} \cdot \frac{p_t}{p_{t-1}}$.
Remove the SFCs that exceed the latency requirement, remove the nodes and links that exceed the resource capacity, in addition, when removing a link, remove the nodes at both ends of the link simultaneously to obtain the set of unmapped nodes $\left\{F'_{s',i,j}\right\}$ and the set of unmapped links $\left\{E'_{s',i,j}\right\}$;
for each $F_{s',i,j}$ in $n_{s',i}$ **do**
  if VNF $F_{s',i,j}$ in $\left\{F'_{s,i,j}\right\}$ **then**
    the sustaining server of VNF $F_{s',i,j-1}$ $K_{s',i,j-1}$ has the highest priority;
    if $K_{s',i,j-1}$ is an edge server **then**
      add the remaining available edge servers to the list l , and sort them in descending order of CPU resource utilization;
    else
      the only node that enters the list l is $K_{s',i,j-1}$;
    end if
    if $K_{s',i,j-1}$ is an edge server **then**
      for each edge server k in list l **do**
        map to the edge server k ;
        if it is successful **then**
          go to step 21;
        else
          go to step 35;
        end if
      end for
    end if
    map the link between VNF $F_{s',i,j}$ and VNF $F_{s',i,j-1}$ to the physical path $\left\{z^{p,q}_{s',i,j}\right\}$;
    if it is successful **then**
      go to step 27;
    else
      go to step 35;
    end if
    if $j = n_{s',i}$ **then**
      map the link between VNF $F_{s',i,j}$ and the destination node V^d $\left\{z^{p,q}_{s',i,n_{s',i}+1}\right\}$;
      if it is successful **then**
        go to step 39;
      else
        go to step 35;
      end if
    end if
    remove the mapping of VNF $F_{s',i,j}$ on the edge server $K_{s',i,j}$;
    if the link between VNF $F_{s',i,j}$ and VNF $F_{s',i,j-1}$ is successfully mapped **then**
      remove the physical path $\left\{z^{p,q}_{s',i,j}\right\}$;
    end if
    if both the nodes and the links are successfully mapped **then**
      break;
    else
      continue;
    end if
    if the mapping of the node is failed **then**
      go to step 50;
    else
      continue ;
    end if
  else
    map to cloud server $K_{s',i,j-1}$,
    if it is successful **then**
      continue;
    end if
    remove all of the mapping in the SFC;
    remove the records in $\left\{F_{s,i,j}\right\}$ and $\left\{E'_{s',i,j}\right\}$;
  end if
end for
for each SFC **do**

 doi:10.20944/preprints202407.1484.v1

if mapping of the SFC is failed **then**
    Call First Fit Algorithm of SFCD (Algorithm 1);
end if
end for
return the migrated placements $\left\{x^{k}_{s',i,j}\right\}$ and $\left\{z^{p,q}_{s',i,j}\right\}$;

4.2.3. FL-Based Algorithms of Network Traffic Prediction

The neural networks in the network traffic prediction process use LSTM as their architecture. The loss function is defined as

$$L(\theta_i) = MSE(x,y) + myloss(x,y) + \ell(x,y) \tag{14}$$

where $MSE(x,y)$ is the mean square error which is calculated by predicted network traffic y and actual network traffic x ; $myloss(x,y)$ is described in Equation (15), and it is based on the proof of the loss function for this problem in our paper [26]; and $\ell(x,y)$ is described in Equation (12).

However, if the migration effect of each node needs to be measured using the migration results of the actual network, the training processes of the models will be prolonged. For instance, the federated cloud server will take a long time to learn for the local model of network traffic prediction, since the migration duration is longer, especially when the scale of the physical network is large. If the migration process during the calculation of the loss is removed, and only the predicted and actual network traffic are used, it will reduce training duration by a significant amount. However, it is no interaction with the physical network, it means that there is no way to know the impact of the predicted network traffic on the network. From the previous analysis, when the predicted network traffic is larger than the actual network traffic, it will take up more resources while no passive migration; when the predicted network traffic is smaller than the actual network traffic, there is a high probability that the resources allocated in advance are not enough, which will lead to passive migration of the SFC. Therefore, higher network traffic prediction values have less adverse effects on NFV networks compared to lower network traffic prediction values. To this end, we first design an FL-based algorithm that uses SFCM to calculate loss function. Then, to further reduce the training duration of the models, we analyze the relationship between the network traffic prediction value and factors such as SFCM and energy consumption, and design an asymmetric loss function computation formula. This loss function does not need to migrate the SFC and greatly reduces the training duration. Define the loss function as follows:

$$myloss(x,y) = \frac{1}{D \times S} \cdot \sum_{i=1}^{D} \sum_{j=1}^{S} \left(l_{i,j}\right)^2 \tag{15}$$

where D is the number of days; S is the number of time slices in one day; and $l_{i,j}$ is defined as follows:

$$l_{i,j} = \sigma \cdot (y-x) \cdot I(y-x) + \omega \cdot (y-x)^2 \cdot I(x-y) \tag{16}$$

where $I(x) = 0$ when $x < 0$; and $I(x) = 1$ when $x > 0$. σ and ω are constant coefficients.

To facilitate understanding, the pseudocode of FL-based training for network traffic prediction (Newloss_FL_NTP) has been listed in Algorithm 3.

Algorithm 3 FL-based Training for network traffic prediction (Newloss_FL_NTP)

Input: Initial models θ_0 , θ_1 ,..., $\theta_{Q'}$.
Output: Improved models θ_0 , θ_1 ,..., $\theta_{Q'}$.
Data: Initial global model θ_0 , history traffic data of each local service provider .
for epoch j **do**
    #Cloud server does
    Send θ_0 to each local model θ_i ;
    #Local edge server does
    for each local model θ_i of each service provider **do**
        Accept and compute loss for local model θ_i according to Equation Equation (14);
        Update parameters of each local model θ_i ;
    end for
    #Cloud server does

 doi:10.20944/preprints202407.1484.v1

Gather parameters of each local model θ_i ;
$\theta_0 \leftarrow \frac{\sum_i \theta_i}{Q'}$
end for

Line 3 of Algorithm 3 introduces ① global model propagation. Line 6 of Algorithm 3 refers to ② traffic predict and ③ compute loss. Line 7 of Algorithm 3 gives ④ update local model for each network service provider. Line 10-11 of Algorithm 3 describe the process of ⑤ local model propagation and ⑥ model aggregation.

5. Experimental Evaluation and Results

In this section, we conduct extensive simulations on a computer with Intel Core I5-8250U 1.8 ghz CPU and 8 GB memory to evaluate the performance of Newloss_FL_NTP algorithm. First, we describe the experiment setup and then present our evaluation metrics. Finally, we illustrate the results of performance comparisions among Newloss_FL_NTP and two related approaches [27,28].

5.1. Simulation Setup

The simulation experiments are executed using Pytorch for neural network implementation. These networks comprise two LSTM layers, each with 50 neurons. The partial parameters are listed in Table 3.

Table 3. Parameter settings.

Hyperparameters	Value
Learning rate	0.0001
Dropout factor	0.2
Number of Service Providers	12
Number of layers	2
Number of neurons each layer	50
Number of Time slices in one day	24
Number of network service providers	12
e_0 in Equation (7)	0.2
e_1 in Equation (7)	0.8
α in Equation (12)	0.1
β in Equation (12)	0.4
γ in Equation (12)	0.5
σ in Equation (16)	0.1
ω in Equation (16)	0.9

The network topology used for simulation experiment is the same with [29], and the network has 200 servers and 991 links. The CPU capabilities of edge servers range from 7 to 10 Cores, the link bandwidth capabilities between two edge servers range from 4 to 10 Mbps, and the link latency between two edge servers range from 1 to 4 s, respectively. The CPU capabilities of cloud servers range from 140 to 200 Cores, the link bandwidth capabilities between two edge servers range from 20 to 50 Mbps, and the link latency between cloud and edge server range from 15 to 60 s, respectively. In addition, it is randomized to assign the network service provider to each cloud server.

Our simulations employ 8 different VNF types and a maximum SFC length of 6. SFCs are randomly generated between various source-destination host pairs, comprising VNFs selected from the aforementioned 8 types. Because the processing time is mainly relevant to the packet header, we do not consider the change of processing times according to the dynamic data flow amount. Besides, the processing latency of edge servers is longer than that of cloud servers. Therefore, the processing latency of edge servers are set from 10 to 20 s, the processing latency of cloud servers are set from 1 to 2 s. Each VNF requires CPU capacities ranging from 1 to 2 Cores, and virtual links require bandwidth

 doi:10.20944/preprints202407.1484.v1

capacity varying from 10 to 20 bps. The SFCs latency limit range from 200 to 280 s. The average performance results are collected from 10 simulation runs in the scenario.

To qualitatively execute evaluations, in addition to our schemes, we have also implemented two typical solutions:

LSTM: It is a network traffic prediction scheme based on LSTM [27].

MSE_Loss_FL: It is a network traffic prediction scheme based on FL, and it uses MSE to compute the loss [28].

5.2. Evaluation Metrics

There are four evaluation metrics, i.e., loss function value, migrated nodes, energy consumption, and failed requests, to investigate the performance of our scheme.

Loss function values: The loss function $L(\theta_i)$ defined in Equation (14) which indicates the convergence performance of the local model in our scheme. The smaller the loss function value is, the better estimation performance a local model has.

Migrated nodes: This metric is referred to as the number of migrated nodes, defined in Equation Equation (10).

For greater clarity, the experimental results are shown as the sum of the number of migrations over multiple iterations, defined as follows:

$$M_t^{sum} = \sum_{i=1}^{t} M_i \tag{17}$$

where t is the total number of iterations of network traffic that have been predicted, and M_i is the number of nodes migrated from the physical network obtained using the ith predicted value of network traffic.

Energy Consumption: This metric is referred to as the total consumed energy to host all VNFs in the physical network , defined in Equation (8). The experimental results shown in the graph are the energy consumption averages over multiple migration iterations, defined as follows:

$$E_t^{Ave} = \frac{\sum_{i=1}^{t} E_i}{t} \tag{18}$$

where E_i is the energy consumption of the physical network obtained using the ith predicted value of network traffic.

Failed Requests: This metric is referred to as the number of the failed SFC requests as defined in Equation (11). The experimental result graph shows the sum of failed migration SFCs over multiple iterations, defined as follows:

$$F_t^{sum} = \sum_{i=1}^{t} F_i \tag{19}$$

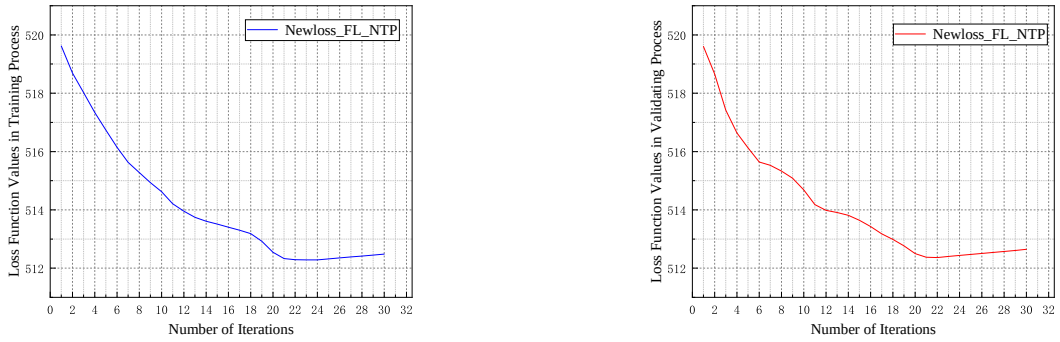
where F_i is the number of failed SFC requests using the ith predicted value of network traffic.

5.3. Simulation Results

1) Convergence of our scheme in Training and Validating processes of local model

Figure 3a,b illustrate the convergence of Newloss_FL_NTP, in the form of loss function values in training and validating processes. In Figure 3a, the loss values of Newloss_FL_NTP gradually decrease during training in the simulation network. After about 20 iterations of training, the loss value decreases to a lesser extent. This indicates that it has reached the convergence range of local learning. Therefore, Newloss_FL_NTP can achieve efficient model training.

 doi:10.20944/preprints202407.1484.v1

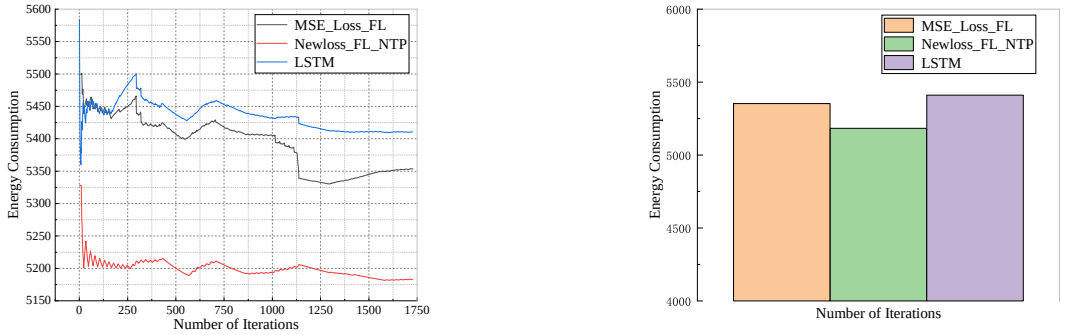


(**a**) Trainning Process. (**b**) Validating Process.

Figure 3. Loss Function Values.

2) Comparison of Energy Consumption, Migrated nodes and Failed SFC requests during performance

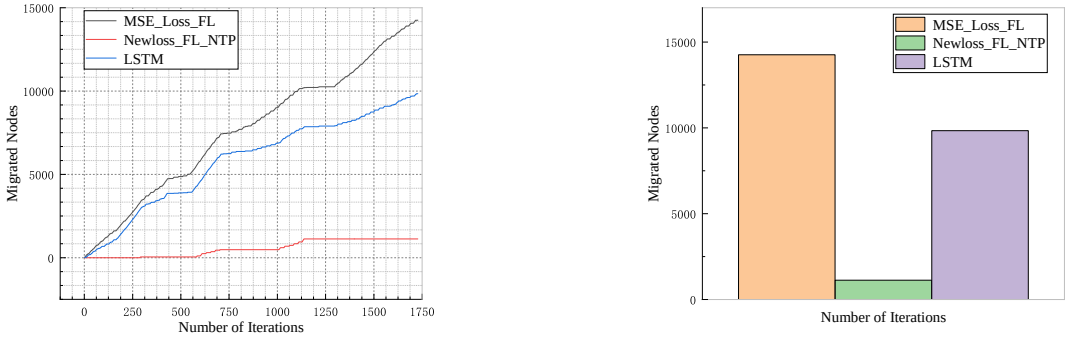
The comparison of our proposed algorithm Newloss_FL_NTP with the MSE_Loss_FL and LSTM algorithms are shown in Figure 4, Figure 5 and Figure 6. Figure 4a and Figure 4b illustrate the energy consumption obtained by Newloss_FL_NTP, MSE_Loss_FL and LSTM. In Figure 5a and Figure 5b, our algorithm's performance in migrated nodes is compared with that of MSE_Loss_FL and LSTM algorithms. Figure 6a and Figure 6b provide valuable insights into the Failed SFC requests associated with MSE_Loss_FL and LSTM algorithm. Notably, our algorithm generally exhibits the best when compared to both the MSE_Loss_FL and LSTM algorithms. The main reason is that they have different loss functions, the algorithms LSTM and MSE_Loss_FL used for comparison directly use the difference between the predicted network traffic and the actual network traffic to calculate the loss value, whereas the algorithm in this manuscript, Newloss_FL_NTP, uses the SFC migration effect of the NFV-based edge-cloud continuum network as the basis for the calculation of the loss function as shown in Equation (14). In the loss function calculation of Equation (14), factors such as the number of failed SFC requests, the number of migrated nodes and energy consumption are included. Therefore, the neural network model trained on the basis of these three factors has better application results in these three aspects.



(**a**) Average Energy Consumption (**b**) Total Energy Consumption

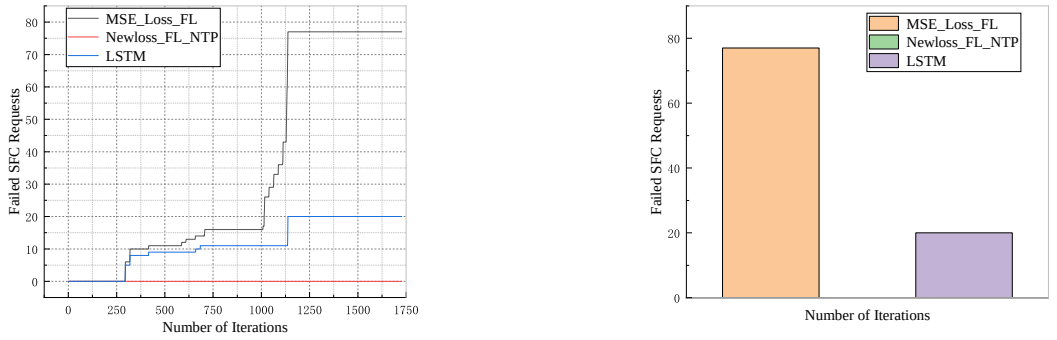
Figure 4. Comparison in Energy Consumption

 doi:10.20944/preprints202407.1484.v1



(**a**) Migrated Nodes (**b**) Total Number of Migrated Nodes

Figure 5. Comparison in Migrated Nodes



(**a**) Failed SFC Requests (**b**) Total Number of Failed SFC Requests

Figure 6. Comparison in Failed SFC Requests

6. Conclusion

This manuscript introduced a novel migration-efficient network traffic prediction algorithm in NFV-based edge-cloud continuum network for multiple network service providers. First, the SFC migration problem was formulated as a mathematical model, and a heuristic solution was proposed which provides available resource while ensures end-to-end delay requirements. And then, the an FL algorithm with a loss function defined by factors such as migration effects and the gap between predicted and actual network traffic. Our proposed algorithm was evaluated through extensive simulations, which showed that it outperforms state-of-the-art schemes (which uses only the difference between predicted and actual network traffic as the loss function), reducing energy consumption by up to 3.29% and 4.39%, reducing the number of migrated nodes by up to 7 times and 11 times, and achieving higher SFC acceptance rates while guaranteeing the end-to-end delay requirement of SFC requests.

The key reason for the better network traffic prediction results in the algorithms of this manuscript is that the feedback from the environment is added to the loss function. Therefore, our future research will aim to further deepen the integration of environmental feedback into the network traffic prediction process. In addition, this research opens avenues for future work, such as developing resource migration strategies for multiple network service providers.

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

Funding: This work is partially supported by the Henan Provincial Department of Science and Technology Program (No. 242102210204).

 doi:10.20944/preprints202407.1484.v1

Data Availability Statement: Not applicable.

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 conflict of interest.

References

1. Gonzalez, A.J.; Nencioni, G.; Kamisiński, A.; Helvik, B.E.; Heegaard, P.E. Dependability of the NFV Orchestrator: State of the Art and Research Challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3307–3329. https://doi.org/10.1109/COMST.2018.2830648
2. Huang, H.; Tian, J.; Min, G.; Yin, H.; Zeng, C.; Zhao, Y.; Wu, D.O. Parallel Placement of Virtualized Network Functions via Federated Deep Reinforcement Learning. *IEEE/ACM Trans. Netw.* **2024**, pp. 1–14. https://doi.org/10.1109/TNET.2024.3366950
3. Yang, L.; Jia, J.; Lin, H.; Cao, J. Reliable Dynamic Service Chain Scheduling in 5G Networks. *IEEE Trans. Mob. Comput.* **2023**, *22*, 4898–4911. https://doi.org/10.1109/TMC.2022.3157312
4. Barbuto, V.; Savaglio, C.; Chen, M.; Fortino, G. Disclosing Edge Intelligence: A Systematic Meta-Survey. *Big Data Cogn. Comput.* **2023**, *7*. https://doi.org/10.3390/bdcc7010044
5. Dustdar, S.; Pujol, V.C.; Donta, P.K. On Distributed Computing Continuum Systems. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 4092–4105. https://doi.org/10.1109/TKDE.2022.3142856.
6. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. https://doi.org/10.1109/JIOT.2016.2579198.
7. Yu, F.; Xu, Z.; Yang, F.; Du, S. Neural Network-Based Traffic Prediction Model with Adaptive Spatial-Temporal Analysis in NFV Networks. 2021 IEEE 21st International Conference on Communication Technology (ICCT), 2021, pp. 502–507. https://doi.org/10.1109/ICCT52962.2021.9657896.
8. Rankothge, W.; Gamage, N.; Dewwiman, H.; Ariyawansa, M.; Suhail, S.; Senevirathne, M. Network Traffic Prediction for a Software Defined Network based Virtualized Network Functions Platform. 2021 6th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2021, Vol. 6, pp. 1–4. https://doi.org/10.1109/ICRAIE52900.2021.9703803.
9. Alawe, I.; Ksentini, A.; Hadjadj-Aoul, Y.; Bertin, P. Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach. *IEEE Netw.* **2018**, *32*, 42–49. https://doi.org/10.1109/MNET.2018.1800104.
10. Spandana, C.; Srisurya, I.V.; A R, P.; S, K.; Sridevi, S.; R, P.K. Application of Machine Learning and Deep Learning Algorithms in Predicting Virtual Network Functions for Network Function Virtualization. 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2023, pp. 1–6. https://doi.org/10.1109/ICCCNT56998.2023.10308121.
11. Eramo, V.; Lavacca, F.G.; Catena, T.; Giorgio, F.D. Reconfiguration of Optical-NFV Network Architectures Based on Cloud Resource Allocation and QoS Degradation Cost-Aware Prediction Techniques. *IEEE Access* **2020**, *8*, 200834–200850. https://doi.org/10.1109/ACCESS.2020.3035749.
12. Zaman, Z.; Rahman, S.; Naznin, M. Novel Approaches for VNF Requirement Prediction Using DNN and LSTM. 2019 IEEE Global Communications Conference (GLOBECOM), 2019, pp. 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9014320.
13. Rahman, S.; Ahmed, T.; Huynh, M.; Tornatore, M.; Mukherjee, B. Auto-Scaling Network Service Chains Using Machine Learning and Negotiation Game. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 1322–1336. https://doi.org/10.1109/TNSM.2020.2995900.
14. Ali, K.; Jammal, M. ML-Based Dynamic Scaling and Traffic Forecasting for 5G O-RAN. 2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2023, pp. 0444–0451. https://doi.org/10.1109/DASC/PiCom/CBDCom/Cy59711.2023.10361376.
15. Bittar, A.; Huang, C. A Vision For Hierarchical Federated Learning in Dynamic Service Chaining. 2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2022, pp. 103–107. https://doi.org/10.1109/NFV-SDN56302.2022.9974900.

16. Fei, X.; Liu, F.; Xu, H.; Jin, H. Adaptive VNF Scaling and Flow Routing with Proactive Demand Prediction. IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 486–494. https://doi.org/10.1109/INFOCOM.2018.8486320.
17. Jalodia, N.; Henna, S.; Davy, A. Deep Reinforcement Learning for Topology-Aware VNF Resource Prediction in NFV Environments. 2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2019, pp. 1–5. https://doi.org/10.1109/NFV-SDN47374.2019.9040154.
18. Carpio, F.; Bziuk, W.; Jukan, A. Scaling migrations and replications of Virtual Network Functions based on network traffic forecasting. *Comput. Netw.* **2022**, *203*, 108582. https://doi.org/https://doi.org/10.1016/j.comnet.2021.108582.
19. Goścień, R. Traffic-aware service relocation in software-defined and intent-based elastic optical networks. *Comput. Netw.* **2023**, *225*, 109660. https://doi.org/https://doi.org/10.1016/j.comnet.2023.109660.
20. Adanza, D.; Gifre, L.; Alemany, P.; Fernández-Palacios, J.P.; de Dios, O.G.; Muñoz, R.; Vilalta, R. Enabling traffic forecasting with cloud-native SDN controller in transport networks. *Comput. Netw.* **2024**, *250*, 110565. https://doi.org/https://doi.org/10.1016/j.comnet.2024.110565.
21. Rajagopal, S.M.; M., S.; Buyya, R. FedSDM: Federated learning based smart decision making module for ECG data in IoT integrated Edge-Fog-Cloud computing environments. *Internet Things* **2023**, *22*, 100784. https://doi.org/https://doi.org/10.1016/j.iot.2023.100784.
22. Parra-Ullauri, J.M.; Madhukumar, H.; Nicolaescu, A.C.; Zhang, X.; Bravalheri, A.; Hussain, R.; Vasilakos, X.; Nejabati, R.; Simeonidou, D. kubeFlower: A privacy-preserving framework for Kubernetes-based federated learning in cloud-edge environments. *Future Gener. Comput. Syst.* **2024**, *157*, 558–572. https://doi.org/https://doi.org/10.1016/j.future.2024.03.041.
23. Mao, Y.; Shang, X.; Liu, Y.; Yang, Y. Joint Virtual Network Function Placement and Flow Routing in Edge-Cloud Continuum. *IEEE Trans. Comput.* **2024**, *73*, 872–886. https://doi.org/10.1109/TC.2023.3347671.
24. Son, J.; Buyya, R. Latency-aware Virtualized Network Function provisioning for distributed edge clouds. *J. Syst. Softw.* **2019**, *152*, 24–31. https://doi.org/https://doi.org/10.1016/j.jss.2019.02.030.
25. Martin-Perez, J.; Malandrino, F.; Chiasserini, C.F.; Bernardos, C.J. OKpi: All-KPI Network Slicing Through Efficient Resource Allocation. IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 804–813. https://doi.org/10.1109/INFOCOM41043.2020.9155263.
26. Hu, Y.; Zhu, L. Migration and Energy Aware Network Traffic Prediction Method Based on LSTM in NFV Environment. *KSII Trans. Internet Inf. Syst.* **2023**, *17*, 896–915.
27. Azzouni, A.; Pujolle, G. A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction. *arXiv* **2017**, arXiv:1705.05690.
28. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19.
29. Hu, Y.; Min, G.; Li, J.; Li, Z.; Cai, Z.; Zhang, J. VNF Migration in Digital Twin Network for NFV Environment. *Electronics* **2023**, *12*. https://doi.org/10.3390/electronics12204324.