This article has been accepted for inclusion in a future issue of this magazine. Content is final as presented, with the exception of pagination.

ACCEPTED FROM OPEN CALL

# Proactive Failure Recovery for NFV in Distributed Edge Computing

Huawei Huang and Song Guo

The authors review the most recent work on the topic of applying NFV to edge networks. They identify that an urgent research challenge is to provide proactive failure recovery mechanism for the NFV-enabled distributed edge computing. To address this issue, they propose a novel management architecture that supports the proactive failover mechanism while provisioning NFV services in distributed edge computing.

## Abstract

Deploying NFV technologies to the edge networks has attracted growing attention in the state-of-the-art studies. In this article, we first review the most recent work on the topic of applying NFV to edge networks. Then, we identify that an urgent research challenge is to provide proactive failure recovery (shorten as failover hereafter) mechanism for the NFV-enabled distributed edge computing. To address this issue, we propose a novel management architecture that supports the proactive failover mechanism while provisioning NFV services in distributed edge computing. Simulation results show that the proposed proactive failover mechanism outperforms the reactive manner significantly in terms of latency spending on failover operations. We hope this article can spur deeper studies on the proactive intelligent resilience mechanism for deploying NFV in distributed edge computing and other related edge intelligence topics.

## Introduction

Nowadays, many emerging mobile multimedia services become pervasive in our daily life. For example, the demands of virtual reality (VR), augmented reality (AR) and ultra high-definition videos keep growing. It can be envisioned that the traffic volume of those applications will have exponential growth when the fifth-generation (5G) communication technology becomes reality.

These services pose very severe challenges to the conventional remote-datacenter based service architecture. Operators and experts have agreed that leveraging edge computing can bring the service closer to the edge of their networks, thus reducing the critical backhaul costs and addressing the stringent low latency requirement of 5G technology. Therefore, edge computing has attracted much attention, since it deploys resources closer to users. Thus, it can provide much better quality of service (QoS) and quality of experience (QoE) than the conventional datacenter based computing paradigm.

Furthermore, the explosive growth of the Internet of Things (IoT) applications has spawned the prosperity of applying Network Function Virtualization (NFV) [1] in edge computing. NFV has received enormous research efforts in recent years, because it decouples network functions from the underlying dedicated hardware, making them operate in the standard virtualization platform as softwares.

In this article, we first summarize the existing work devoted to the NFV provisioning in edge networks. Via the review, we find out that most existing studies adopt the reactive failover mechanism, and the proactive resilience management for NFV in distributed edge computing has been in urgent need. We believe that the challenge of providing the proactive failover mechanism is the prediction of potential failures occurring in NFV environments of distributed edge computing. To this end, we propose a proactive failover architecture that can predict failures based on machine learning algorithms, and thus enable the proactive failover mechanism for NFV in distributed edge computing. Simulation results demonstrate that the proactive failover mechanism shows overwhelming advantages compared to the reactive approach in terms of failover latency.

## State-of-the-Art NFV Technologies and Resilience Issues in Edge Computing

### NFV-enabled Applications in Edge Computing

We first review the most recent studies that aim at deploying NFV to edge computing. For example, Li *et al.* [2] devised an NFV based platform for mobile edge computing (MEC) that can analyze the QoS and QoE of services offered by the edge cloud. Through the QoE measurement toward a service of HTTP video deployed in edge servers, it can be found that the proposed NFV-based MEC platform outperforms remote servers in terms of higher bandwidth, lower service latency and packet loss. Based on the flexibility of placing MEC services brought by NFV technologies, Yang *et al.* [3] studied the placement and redistribution of NFV resources for mobile multimedia applications to ensure the low latency requirement. In particular, the proposed scheme considers the effects of auto-scaling and load balancing when allocating resources dynamically. To enable third-party applications to deploy Virtual Network Functions (VNFs) for their customized requirements, Boubendir *et al.* [4] presented an on-demand NFV management model that offers network application programming interfaces (APIs) to network operators for the deployment of network functions. An OPNFV platform based on Docker containers has been implemented to explore the features to serve the web real-time communication applications in edge. To offer an integrated on-demand infrastructure for orchestrating NFV resources in mobile edge computing nodes, Carella *et al.* [5] presented

Digital Object Identifier:
10.1109/MCOM.2019.1701366

*Huawei Huang is with Sun Yat-Sen University; Song Guo is with The Hong Kong Polytechnic University.*



0163-6804/19/$25.00 © 2019 IEEE

This article has been accepted for inclusion in a future issue of this magazine. Content is final as presented, with the exception of pagination.

| Literature | Application orientated | Virtualization tools | Main objective, QoS/QoE | Featured approaches |
|---|---|---|---|---|
| Boubendir *et al.* [4] | Web real-time communication | Docker container | To minimize service time | On-demand deployment of VNFs |
| Yang *et al.* [3] | Mobile multimedia, applications | VMs | To minimize response time | Auto-scaling and load-balancing |
| Li *et al.* [2] | HTTP video | OPNFV | To lower latency, jitter and packet loss rate | NFV-based MEC platform |
| Carella *et al.* [5] | 5G ready multi-access edge computing | Container, Openstack | To provide on-demand integrated resource orchestration solution | Open baton MANO framework |
| Nam *et al.* [6] | Radio access networks, MEC services | VMs | To improve hit rate of VNF, to minimize end-to-end service time | Finding the optimal number of MECs based on popularity of VNFs |
| Leconte *et al.* [7] | Network slicing in distributed cloud | Adaptive to cloud-native container | To compute the best resource provisioning for network slices | Proposing an ADMM algorithm to allow auto-scaled slice provisioning |
| Lingen *et al.* [8] | IoT services, carrier-grade assurance in IoT | MANO framework | To offer a uniform management architecture for IoT services | A model-driven service-centric approach |
| Fawcett *et al.* [9] | Content delivery network as a case study | Docker engine | To find the optimal placement of services in distributed Fog | Proposing Siren as a new prototype to deploy VNFs in Fog |

**Table 1.** Comparisons of studies related to deploying NFV in edge computing.

Nowadays, many emerging mobile multimedia services become pervasive in our daily life. For example, the demands of virtual reality (VR), augmented reality (AR) and ultra high-definition videos keep growing. It can be envisioned that the traffic volume of those applications will have exponential growth when the 5G communication technology becomes reality.

a prototyping architecture based on the open baton (https://openbaton.github.io/) management and orchestration (MANO) framework that has been standardized by the European Telecommunications Standards Institute (ETSI), and could deploy container-based network services in 5G ready networks.

By applying NFV Technologies to mobile edge computing, Nam *et al.* [6] studied a clustered NFV service chaining scheme to minimize the end-to-end service time in radio access networks (RANs). For 5G communications, it is believed that *network slicing* plays a critical role during the allocation of computing functionalities over the edge cloud. To allocate the distributed slice resources considering the efficiency of both traffic-fairness and computing-fairness, Leconte *et al.* [7] proposed an iterative algorithm based on the alternating direction method of multipliers (ADMM). Under the combined network environment consisting of SDN and NFV technologies, the proposed algorithm enables auto-scaled real-time provisioning for network slices. Based on the ETSI's MANO architecture, Lingen *et al.* [8] envisioned an unavoidable convergence of NFV, 5G and fog computing. Specifically, the converged architecture could offer IoT services from cloud to edge. Fawcett *et al.* [9] presented a prototype platform, named Siren, as a showcase to deploy VNFs to edge computing in a distributed manner.

We also summarize the features of each work that promotes the NFV in edge computing. The details of comparison are shown in Table 1.

### Proactive Provisioning for NFV-enabled Networks

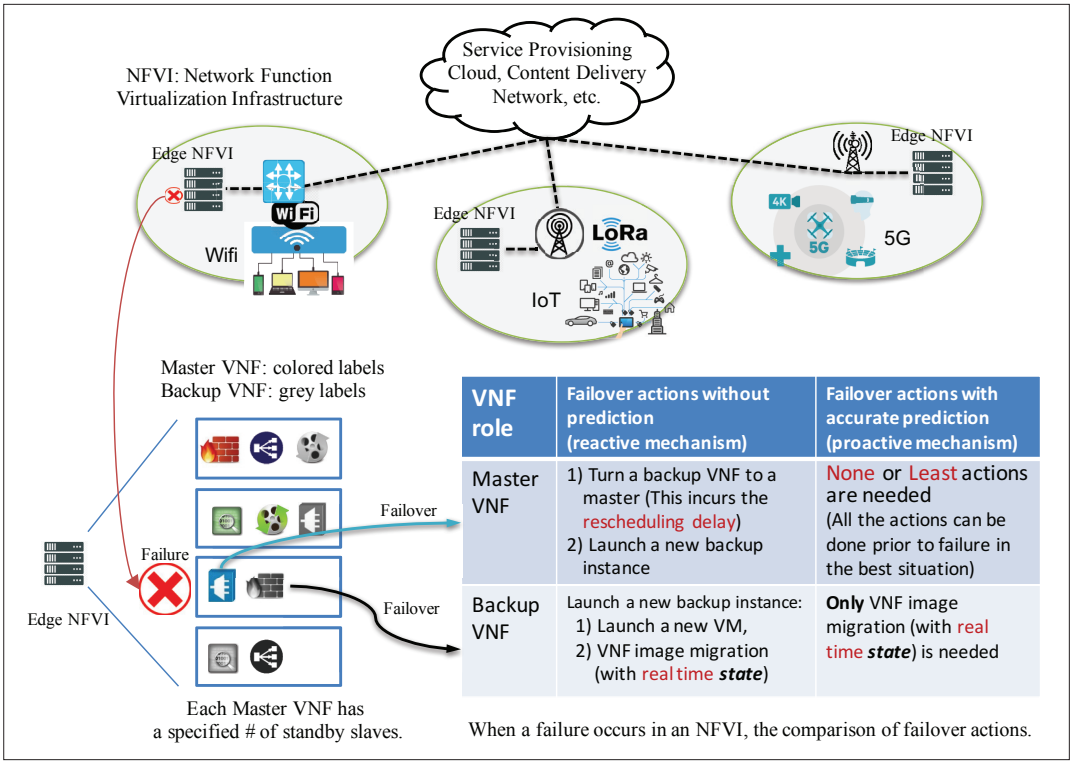
We also notice that the resilience issues of NFV have attracted growing attention. We take three state-of-the-art studies as examples to show this research interest. At first, each VNF instance has an availability probability because of some network failures such as failures in servers or links. Therefore, each service chain also has an availability. To achieve high reliability, Fan *et al.* [10] adopt the traditional active standby redundancy policy, where each VNF has a backup instance such that the primary VNF can be recovered by using the standby instance in case it fails. For example, different redundancy deployment strategies indicate different availabilities of service chains. The main idea of this article is to find the optimal deployment of service chains while guaranteeing a certain degree of availability of each service chain. In the second work, Zhang *et al.* [11] propose a more practical proactive approach for VNF deployment given the time overhead. In particular, proactive provisioning is achieved by a two-phase approach: predicting upcoming traffic demand for each service chain, and reserving virtual machines (VMs) to deploy VNF instances. Sciancalepore *et al.* [12] also mention that the allocation of 5G network resources can benefit from traffic analysis and prediction.

Through the review of the state-of-the-art work, we find that the proactive failover mechanism for deploying NFV technologies to the distributed edge computing is still missing. Furthermore, such a proactive mechanism in distributed edge networks owns its unique characteristics that are different from those in other networks. We will elaborate the difference in the next section. Thus, to this end, we study such a mechanism in this article to meet the urgent need.

## Our Proposal: Proactive VNF Failover Based on Failure Prediction

To improve the resilience of NFV services deployed in distributed edge networks, we propose the proactive failover mechanism in this section. First we compare the differences between

For a failed master VNF instance, all the failover operations or most of them including turning a backup VNF to a master, and launching a new backup instance can be conducted prior to the failure moment if the failures can be actually predicted. Therefore, nothing or only the least failover actions need to be executed when a failure event occurs in an edge server.




| VNF role | Failover actions without prediction (reactive mechanism) | Failover actions with accurate prediction (proactive mechanism) |
|---|---|---|
| Master VNF | 1) Turn a backup VNF to a master (This incurs the rescheduling delay)<br>2) Launch a new backup instance | None or Least actions are needed<br>(All the actions can be done prior to failure in the best situation) |
| Backup VNF | Launch a new backup instance:<br>1) Launch a new VM,<br>2) VNF image migration (with real time ***state***) | **Only** VNF image migration (with real time ***state***) is needed |

When a failure occurs in an NFVI, the comparison of failover actions.

**Figure 1.** When a failure occurs in an edge NFVI server, the failover should be conducted to both master and backup VNFs.

the proactive and reactive failover operations for provisioning NFV services in distributed edge networks. We then elaborate the proposed architecture that supports the proactive failover mechanism.

## Proactive and Reactive Failover Mechanisms

In Fig. 1, we compare the difference between the proactive and reactive failover mechanisms. As shown, once an edge server residing in the network function virtualization infrastructure (NFVI) fails, the administrator needs to conduct the failover for both the master and the backup VNF instances that were running in this failed server. We specify the detailed operations under two such mechanisms as follows.

**Reactive Failover:** For a failed master VNF instance, the first step is to select a backup VNF and upgrade its role from *backup* to *master*. This requires a rescheduling of the routing paths that connect successive network functions in the service chain, by taking advantage of the popular Software-Defined Networking technology. The next step is to launch a new backup instance for this newly selected master VNF.

For a failed backup VNF, the only operation required is to launch another new backup instance. From the table shown in Fig. 1, we can see that launching a new backup instance also includes two steps:

- Launch a new VM container in an edge server.
- Migrate the VNF image with the real-time *state* to the new VM container launched just now.

Note that the *state* refers to some common information shared among VNF instances of a specified network function, for example, activity logs and port mappings [13].

**Proactive Failover:** For a failed master VNF instance, all the failover operations or most of them including turning a backup VNF to a master, and launching a new backup instance can be conducted prior to the failure moment if the failures can be actually predicted. Therefore, nothing or only the least failover actions need to be executed when a failure event occurs in an edge server.

On the other hand, for a failed backup instance, if the launching of a new VM can be completed in advance based on the fact that failures are actually predicted, the only thing that needs to be done is to migrate the VNF image with real-time state to the new VM container. In the best situation under the proactive mechanism, the VM launching delay can be saved completely.

We now compare the differences of delay between these two failover policies.

In the reactive approach, the failover operations incur the *service-disruption-delay* [13], mainly including the following three aspects.

**Flow-Rescheduling Delay:** Incurred by routing path calculation in the controller and installing new forwarding rules to switches. Since turning a backup VNF to a master will lead to rescheduling the routing path between the NFV gateway and the master VNF instance, the reactive failover mechanism incurs the flow-rescheduling delay.

**VM-Launch Delay:** Refers to the time spent launching VM containers for the failed VNF instances.

**VNF-Image Migration Delay:** Indicates the latency caused by the VNF image-migration from an instance of the failed network function to the newly launched VM container.

In contrast, the delay under the proactive approach mainly involves the *VNF-image migration delay*. The reasons are twofold:

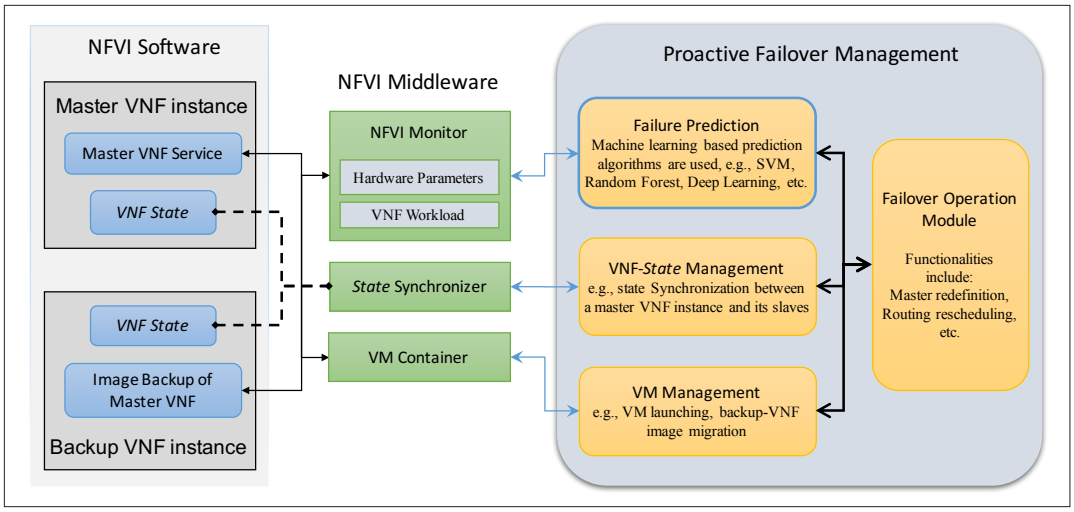



**Figure 2.** An architecture of proactive failover mechanism for NFVI in distributed edge computing. The arrowed lines denote the correlations between components.

The failover operation module provides the core functionalities of the proactive failover mechanism. Based on the supports from the other three modules, we need to define two major functionalities, that is, master VNF redefinition and routing scheduling, to realize the proactive failover mechanism.

- The proactive approach can turn a backup VNF instance to a master prior to a failure event based on prediction. Thus, the flow-rescheduling delay can be avoided completely or partially.
- The VM-launch can be completed entirely or partially in advance based on the failure prediction results.

It is worth noting that we cannot save the *VNF-image migration delay* even under the proactive approach. The reason is that the VNF-image with the real-time *state* deciding exactly when failure occurs is mandatory when migrating a VNF instance [14]. It is meaningless to migrate a VNF-image with an out-of-date state to the new VM container prior to a failure event.

## Proactive NFV Failover Architecture in Distributed Edge

To enable the proactive failover mechanism of provisioning NFV in distributed edge computing, we design a management architecture as shown in Fig. 2, which includes three main layers.

The left-hand-side shows the *NFVI Software* layer, which mainly includes two components, the *master VNF instances* and the *backup VNF instances*. The master VNF service is in charge of handling the specified NFV service. The *VNF state* residing in both the master and backup instances maintains the states while the VNF instances are handling user traffic flows. The *image backup of the master VNF* residing in the *backup VNF instances* stores the real-time standby instance of its corresponding master VNF.

We then explain the right-hand-side layer, that is, *proactive failure-recovery management*. It includes four modules. The most critical and fundamental module is the *failure prediction module*, in which the operator can run various machine learning based prediction algorithms for failure prediction. The predicted results offer the foundation of decision making for proactive failover operations. Failures can be prevented during run-time by learning from past failures. We will present the failure prediction methodology using several machine learning methods, such as supporting vector machine (SVM) and random forest, in the section on evaluation. Based on the failure prediction results, the advantage of proactive failover is also shown in the performance evaluation by comparing the failover delay with the reactive mechanism. The second module is *VNF-state management*, which is in charge of synchronizing states shared among the master and its standby slave VNF instances of a specified network function. The third module is referred as *VM management*, which decides the operations of launching new VM containers for any new VNF instances, and migrating a backup-VNF image to a new location when a new backup instance is needed to satisfy the predefined redundancy requirement. Finally, the *failover operation module* provides the core functionalities of the proactive failover mechanism. Based on the supports from the other three modules, we need to define two major functionalities, that is, *master VNF redefinition* and *routing scheduling*, to realize the proactive failover mechanism. The first functionality designates a backup VNF instance as the new master when a master VNF goes down due to a failure event, while the second functionality calculates the new routing paths after a failure event, to provide routing rescheduling solutions for user traffic flows according to certain service agreements.

The middle part of the figure illustrates the *NFVI middleware* layer, which includes three components: NFVI monitor, state synchronizer and VM container. The critical NFVI monitor is used to monitor the hardware parameters of NFVI server machines, for example, CPU and memory usages and the workload of master VNF instances. The collected log information can be fed to the aforementioned *failure prediction module*, for predicting the server failures. The *state synchronizer* ensures the state synchronization between the master VNF instances and their backup instances. To each *backup VNF instance*, in order to recover the NFV service immediately once a failure damages the master VNF instance, the image of the master VNF instance and the corresponding VNF-state should always be updated and synchronized dynamically. Note that the correlations among the associated components are demonstrated with

This article has been accepted for inclusion in a future issue of this magazine. Content is final as presented, with the exception of pagination.

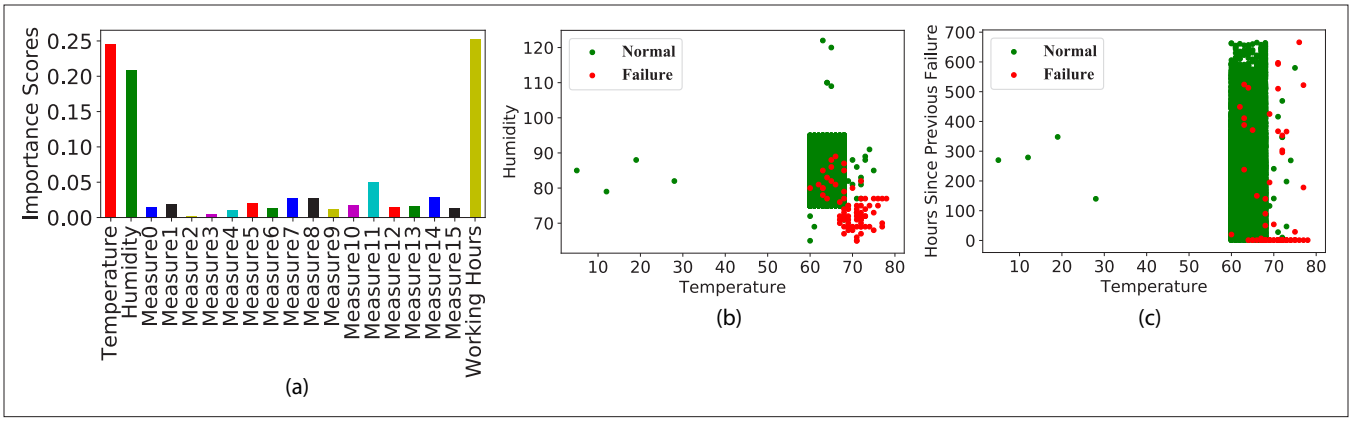



**Figure 3.** The important features of the adopted dataset, and the scatter showing the correlations of two pairs of the important features. The scatter of normal and failure cases illustrates obvious patterns, giving us accurate prediction potential using machine learning based algorithms. a) Importance scores of dataset features; b) humidity vs. temperature; c) temperature vs. working hours.

arrowed lines in Fig. 2. The interaction of each correlation can be implemented by developing dedicated APIs.

## Evaluation

In this section, based on the proposed proactive failover architecture, we compare the delay performance of *proactive* and *reactive* failover mechanisms.

### Simulation Settings

**Traces:** First we generate the trace of fixed inter-server connection delay for rescheduling traffic flows. We consider an edge network with five edge clusters, each of which has 10 servers deployed as an edge NFVI. The lower and upper bounds for generating the connection delay among edges are set to 10 milliseconds (ms) and 1000 ms, respectively. In contrast, the lower and upper bounds for generating the connection delay among servers that are located at a same NFVI are assigned to 10 ms and 100 ms, respectively.

Note that in the simulation, we do not emphasize the design of machine learning based prediction algorithms, which will be studied in our future work. Instead, we use a trace of failure events as the input information for the proactive failover operations.

To generate such a trace, we set the failure probability of each edge server as 0.01 per time slot. Then, we yield a 100 time-slot long trace of failure events. Referring to [15], we set the delay spending on the rescheduling flows after a failure event and the delay of launching a new VM container as 10 ms and 40 ms, respectively.

For service-demand, we create several synthetic demand traces by randomly locating 100 mobile users at edges, and varying the number of network functions in the service chain from one to five for each demand. We refer to this number as the hop-length of a service chain. In particular, we adopt the "1+1" standby redundancy policy for each network function. That is, there is always one backup VNF instance for each master VNF instance deployed in a difference edge server rather than the one where the master is located, in case of edge server failures.

**Metric:** We emphasize the performance metric in terms of the cumulative delay spending on the failover operations under either the proactive or reactive mechanism.

**Algorithms:** To compare the failover delay performance, we combine the *reactive/proactive* policies with two backup VNF-launching algorithms called *greedy* and *random*. As described previously, the migration of a VNF image is always necessary when launching a new backup instance in the failover operation. In the *greedy* algorithm, we select a target edge server that has the minimum connection delay with the image-source server. In contrast, in the *random* algorithm, the target edge server for the destination of VNF image-migration is selected randomly among all available servers.

Thus, we have four variations of failure operation algorithms, which are denoted by "*reactive, greedy*," "*reactive, random*," "*proactive, greedy*" and "*proactive, random*.". Under the *proactive* mechanism, we assume that the failure events are first predicted by exploiting machine learning algorithms. According to the prediction results, we conduct the failover operations.

### Failure Prediction Based on Machine Learning Algorithms

**Prediction Methodology:** Here we show a case-study of failure prediction, which is the prior step for failover operations toward the NFVI servers in distributed edge networks.

***Dataset:*** We find a small-scale trace of server machine failures from an open dataset depository (https://bigml.com/). This dataset is 1.1 megabytes big in size, including 8784 lines of server-parameter records with 19 primary features. Notice that we use this simple dataset only as a showcase to show the methodology of the failure prediction module in the proposed failover architecture. We first analyze the features of this dataset to determine the important factors that lead to failures in the machine. The importance scores of each feature are shown in Fig. 3a. We can observe that the features *temperature*, *humidity* and *hours since previous failures* (referred to as *working hours*) are illustrated as the important factors that will incur failures of server machines. We then analyze the correlations of these important features. The correlations of two pairs of these important features, that is, humidity versus temperature and temperature versus working hours, are shown in Figs. 3b and 3c, respectively. From these two figures, we can see that the scatter

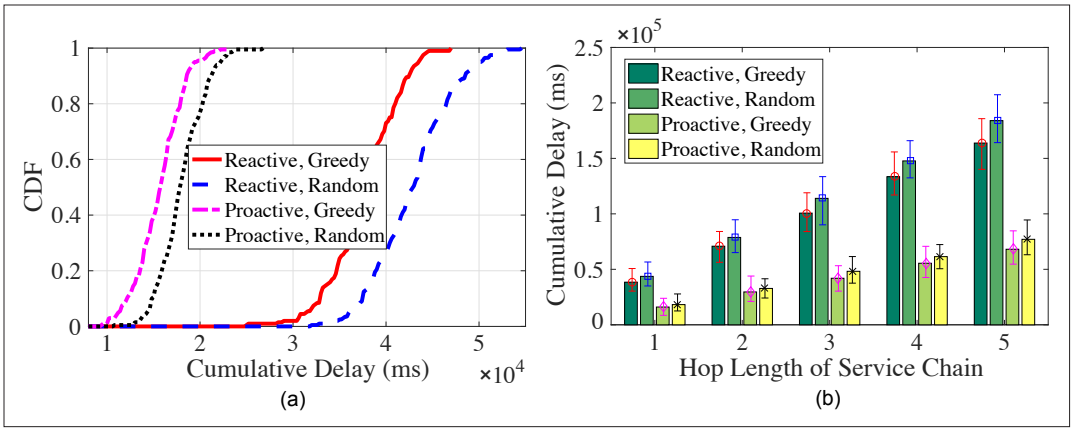



**Figure 4.** The delay performance of the four failover algorithms: a) CDF of delay under four algorithms; b) cumulative delay vs. length of service chain.

of failures and normal cases demonstrates obvious patterns. Thus, we can exploit several classic machine learning algorithms to predict the failures occurred in server machines. It is worth noting that the original dataset is extremely imbalanced since the failure records are too few. In particular, we conduct preprocessing by duplicating the failure records to achieve the balanced number of both normal and failure cases.

***Machine Learning Algorithms:*** Using the processed dataset, we run three machine learning algorithms to provide prediction results. The first machine learning algorithm is called *one-class SVM*, which is an unsupervised learning algorithm. The second one is *SVC* function provided by *scikit learn* lib. Essentially, this function is still a version of SVM. The other algorithm is *random forest*.

Specifically, we shuffle the dataset and then select 70 percent as the training set, the others as the testing set. Then, three prediction models are trained using the training set based on the three machine learning algorithms. Finally, the corresponding prediction accuracies are examined using the testing set.

**Prediction Accuracy:** Next, we present the failure prediction results using three machine learning algorithms based on the dataset aforementioned. Finally, the *one-class SVM* shows the worst performance with average prediction accuracy of 50.08 percent, while SVC and *random forest* are exhibiting 99.99 percent and 99.98 percent failure prediction accuracies, respectively.

### Simulation Results Based on Accurate Failure Prediction

Based on the accurate failure prediction yielded by the *SVC* or *random forest* approaches, we conduct simulations to show the merits of a proactive failover mechanism against the reactive mechanism.

In the first suite of simulation, we consider that only one network function exists in the service chain for each user demand. Figure 4a shows the cumulative deferential function (CDF) of total cumulative delays caused by failure events under different algorithms (i.e., greedy and random) and policies (i.e., reactive and proactive). We can observe two explicit performance perspectives:

- The cumulative delay under the *proactive* policy is much smaller than under the *reactive* policy
- The cumulative delay under the *greedy* algorithm is much lower than under the *random* algorithm.

Next, we evaluate the impact of service-demand scale under different failover mechanisms while varying the length of service chains from 1 to 5. Figure 4b demonstrates the cumulative delay spending on failover operations under the four combined algorithms. First we see the cumulative delay of all combined algorithms illustrates linear increasing when the length of the service chains is growing. Similarly, we can see that the "*proactive, greedy*" and "*reactive, random*" combined algorithms show the best and the worst performance, respectively. Thus, we can conclude that *proactive* policy and *greedy* algorithm exhibit overwhelming advantages in terms of failover delay corresponding to failure events occurred in the NFV provisioning for distributed edge computing.

## Conclusion

In this article, we studied the proactive failover mechanism for NFV services in distributed edge computing. We devised an architecture that supports a proactive failover mechanism based on failure prediction. Its significant advantages have been demonstrated in simulation by comparing the failover latency with the reactive mechanism. It is worth noting that the implementation of this proactive mechanism with many other sophisticated failure prediction algorithms is an open issue that needs further studies.

We devised an architecture that supports a proactive failover mechanism based on failure prediction. Its significant advantages have been demonstrated in simulation by comparing the failover latency with the reactive mechanism. It is worth noting that the implementation of this proactive mechanism with many other sophisticated failure prediction algorithms is an open issue that needs further studies.

## Acknowledgment

This work is partially supported by the Shenzhen Basic Research Funding Scheme under research grant no. JCYJ20170818103849343.

## References

[1] H. Huang *et al.*, "Service Chaining for Hybrid Network Function," *IEEE Trans. Cloud Computing*, 2017, DOI:10.1109/TCC.2017.2721401.

[2] A. Boubendir, E. Bertin, and N. Simoni, "On-Demand, Dynamic and At-The-Edge VNF Deployment Model Application to Web Real-Time Communications," *Proc. 12th Int'l. Conf. Network and Service Management (CNSM)*, 2016, pp. 318–23.

[3] B. Yang *et al.*, "Seamless Support of Low Latency Mobile Applications with NFV-Enabled Mobile Edgecloud," *Proc. IEEE Int'l. Conf. Cloud Networking (Cloudnet)*, 2016, pp. 136–41.

[4] S. Li *et al.*, "QoE Analysis of NFV-Based Mobile Edge Computing Video Application," *Proc. IEEE Int'l. Conf. Network Infrastructure and Digital Content (IC-NIDC)*, 2016, pp. 411–15.

This article has been accepted for inclusion in a future issue of this magazine. Content is final as presented, with the exception of pagination.

[5] G. A. Carella *et al.*, "Prototyping NFV-Based Multi-Access Edge Computing in 5G Ready Networks with Open Baton," *Proc. IEEE Conf. Network Softwarization (NetSoft)*, 2017, pp. 1–4.
[6] Y. Nam, S. Song, and J.-M. Chung, "Clustered NFV Service Chaining Optimization in Mobile Edge Clouds," *IEEE Commun. Lett.*, vol. 21, no. 2, 2017, pp. 350–53.
[7] M. Leconte *et al.*, "A Resource Allocation Framework for Network Slicing," *Proc. IEEE Conf. Computer Commun. INFOCOM*, 2018, pp. 2177–85.
[8] F. van Lingen *et al.*, "The Unavoidable Convergence of NFV, 5G, and Fog: A Model-Driven Approach to Bridge Cloud and Edge," *IEEE Commun. Mag.*, vol. 55, no. 8, 2017, pp. 28–35.
[9] L. Fawcett and N. Race, "Siren: A Platform for Deployment of VNFs in Distributed Infrastructures," *Proc. Symposium on SDN Research. ACM*, 2017, pp. 201–02.
[10] J. Fan *et al.*, "Availability-Aware Mapping of Service Function Chains," *Proc. IEEE Int'l. Conf. Computer Commun. (INFOCOM)*, 2017, pp. 1–9.
[11] X. Zhang *et al.*, "Proactive VNF Provisioning with Multi-Timescale Cloud Resources: Fusing Online Learning and Online Optimization," *Proc. IEEE Int'l. Conf. Computer Commun. (INFOCOM)*, 2017, pp. 1–9.
[12] V. Sciancalepore *et al.*, "Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization," *Proc. IEEE Int'l. Conf. Computer Commun. (INFOCOM)*, 2017, pp. 1–9.
[13] J. Sherry *et al.*, "Rollback-Recovery for Middleboxes," *ACM SIGCOMM Computer Commun. Review*, vol. 45, no. 4, 2015, pp. 227–40.
[14] L. Nobach *et al.*, "Statelet-Based Efficient and Seamless NFV State Transfer," *IEEE Trans/Network and Service Management*, vol. 14, no. 4, 2017, pp. 964–77.
[15] J. Martins *et al.*, "Clickos and the Art of Network Function Virtualization," *Proc. 11th USENIX Conf. Networked Systems Design and Implementation*, USENIX Association, 2014, pp. 459–73.

## Biographies

Huawei Huang [M'16] received his Ph.D. in computer science and engineering from the University of Aizu, Japan. His research interests mainly include software-defined networking (SDN), NFV, and edge computing. He was a visiting scholar at the Hong Kong Polytechnic University from 2017 to 2018. He was a post-doctoral research fellow of JSPS from 2016 to 2018. He was an assistant professor at Kyoto University, Japan, from 2018 to 2019.

Song Guo [M'02, SM'11] received his Ph.D. degree in computer science from the University of Ottawa, Canada. He is currently a full professor in the Department of Computing, The Hong Kong Polytechnic University. His research interests mainly include cloud and green computing, big data, and cyber-physical systems. He serves as an editor of several journals, including IEEE TPDS, TETC, TGCN, and *IEEE Communications Magazine*. He is a senior member of IEEE and ACM, and an IEEE Communications Society Distinguished Lecturer.