AN AUTOMATED PIPELINE FOR ADVANCED FAULT TOLERANCE IN EDGE COMPUTING INFRASTRUCTURES

Theodoros Theodoropoulos
Department of Informatics and Telematics
Harokopio University
Athens, Greece, 16671
ttheod@hua.gr

Antonios Makris
Department of Informatics and Telematics
Harokopio University
Athens, Greece, 16671
amakris@hua.gr

John Violos
Department of Informatics and Telematics
Harokopio University
Athens, Greece, 16671
violos@hua.gr

Konstantinos Tserpes
Department of Informatics and Telematics
Harokopio University
Athens, Greece, 16671
tserpes@hua.gr

August 5, 2022

ABSTRACT

The very fabric of Edge Computing is intertwined with the necessity to be able to orchestrate and manage a huge number of heterogeneous computational resources. On top of that, the rather demanding Quality of Service (QoS) requirements of Internet of Things (IoT) applications that run on these resources, dictate that it is essential to establish robust Fault Tolerance mechanisms. These mechanisms should be able to guarantee that the requirements will be upheld regardless of any potential changes in task production rate. To that end, we suggest an Automated Pipeline for Advanced Fault Tolerance (APAFT) that consists of various components that are designed to operate as functional blocks of an automated closed-control loop. Furthermore, the suggested pipeline is able to carry out the various Horizontal Scaling operations in a proactive manner. These Proactive Scaling capabilities are achieved via the use of a dedicated Deep Learning (DL)-based component that is able to perform multi-step prediction. Our work aims to introduce a number of mechanisms that are able to leverage the benefits that are provided by the multi-step format in a more refined manner. Having access to information regarding multiple future instances allows us to design automated resource orchestration strategies that cater to the specific characteristics of each type of computational node that is part of the Edge Infrastructure.

1 Introduction

Novel technological concepts, such as the Internet of Things (IoT) and extended Reality (XR) [1] applications, are intertwined with a plethora of demanding Quality of Service (QoS) requirements [2]. Both of these paradigms rely heavily on Edge Computing to meet these requirements. Within the context of the IoT and the XR paradigms, Edge computing enables tasks to be processed closer to the devices where they are being generated. By doing so, the overall end-to-end latency is significantly reduced. This inherit characteristic of Edge Computing is of vital importance to these modern paradigms since they both present the need for low end-to-end latency. The definition of end-to-end latency entails the time that is required for a task to be processed upon its arrival at a specified computational node. Therefore, it is extremely important for Edge Computing Infrastructures to have sufficient computational resources to process tasks within the timeframes specified by QoS requirements. If not enough computational resources are allocated, then overheads in task execution are expected to be incurred.

The number of tasks that are being offloaded to the Edge devices is extremely volatile in nature and susceptible to fluctuation caused by various factors. Sudden bursts in task production are a rather unfortunate inevitability. The majority of resource allocation mechanisms are designed to operate based on real-time measurements that correspond to some specific metric. For instance, the Kubernetes Horizontal Pod Autoscaler[1] is designed to perform horizontal scaling based on the ongoing CPU utilization. The main issue with this type of reactive approaches is that in various cases (e.g. Virtual Machines), the process of acquiring additional computational nodes, is not instant and often requires several minutes. As a result, it is of paramount importance to establish robust computing paradigms that are able to withstand the sudden changes in task production without jeopardizing the system's ability to comply with the QoS requirements.

One way of establishing such robust computing paradigms is through the implementation of Proactive Fault Tolerance. Proactive Fault Tolerance is implemented via the incorporation of prediction mechanisms that are able to detect a potential fault before it occurs and to assist in avoiding its influence on the Infrastructure's ability to process tasks. Multi-step predictions provide a wider range of information when compared to single-step predictions.

Modern-day computing paradigms like Cloud Computing and Edge Computing rely heavily on the orchestration and management of multiple clusters of heterogeneous computational resources. This means that each processing node possesses various distinct characteristics that need to be carefully incorporated into any potential attempts at implementing an orchestration strategy. In the context of this paper, these characteristics include the time that is required for a specific computational node to be deployed and the maximum number of tasks per minute a specific type of computational node can handle, without showing any signs of deterioration in its performance.

Finally, another important fact to consider is the number of computational nodes that may be incorporated in Edge Infrastructures. In many cases, the sheer size of these Edge Infrastructures may be prohibitively large in terms of orchestration and human management. Thus, it is important to establish autonomous closed-control loops that are able to carry out the decision making processes in a manner that minimizes the need for human intervention.

These facts are the driving force behind the need to propose An Automated Pipeline for Advanced Fault Tolerance (APAFT) in Edge Computing Infrastructures, which focuses on processing faults. These faults are associated with shortage of resources that may prevent the aforementioned Edge Infrastructures from executing tasks in a manner that is compliant with the QoS requirements. Within this work, faults correspond to recorded end-to-end delays that are greater than one second.

Our primary research goal is to propose a suitable automated pipeline that guarantees advanced Fault Tolerance. The cornerstone of this pipeline is the ability to conduct the appropriate Horizontal scaling operations in a proactive manner. Whenever a potential congestion in task execution is expected to occur, node replication is triggered to mitigate the negative effects that are bound to take place. To that end, several additional mechanisms are required in order to provide specific functionalities that greatly enhance the Fault Tolerance capabilities of the Edge Infrastructure.

The three major contributions of our research are:

- The proposal of an automated pipeline that is able to provide Advanced Proactive Fault Tolerance in Edge Computing Infrastructures.
- The detailed analysis of the various components that serve as functional blocks of the proposed architecture.
- The experimental evaluation of the proposed pipeline in the context of various Fault Tolerance metrics.

The rest of the paper is structured as follows: Section 2 highlights the related work in Fault Tolerance, network traffic prediction, time series and deep learning. Section 3 explains the concept of Proactive Fault Tolerance and provides a detailed analysis of the APAFT architecture. Section 4 describes the experimental evaluation process that was conducted using the CloudSim Plus simulation environment. Section 5 concludes the merits of this work and provides directions in regards to potential future work.

2 Related Work

Fault Tolerance mechanisms can be divided in two distinct categories. The first category corresponds to reactive mechanisms and the second category corresponds to proactive mechanisms. The reactive mechanism are designed to minimize the ramifications that derive from failures in the Edge Infrastructure. These mitigation measures are deployed only after the faults occur. The most notable reactive Fault Tolerance mitigation measures are reactive replication, re-submission and checkpointing. Replication [3] refers to the process of replicating a task to more than one computational nodes. Replication is able to provide advanced reliability of the Infrastructure and greater chances

[1]https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

for each task to be fully processed. Unfortunately, this process requires an excess of resources and thus the overall cost is significantly increased. Resubmission [4] refers to the process of executing a failed task from the start. This process may be repeated on the same computational node or another one that has been chosen. Resubmission requires the rescheduling of task execution and thus presents the drawback of increased response times. Checkpointing [5] refers to the process of creating snapshots during task execution and in the event of task failure, the task resumes its execution process from the last available checkpoint. Contrary to resubmission, checkpointing does not require a rescheduling of task execution but instead requires a dedicated storage in order to maintain the aforementioned snapshots. Furthermore, the execution time is increased since this method also includes the recovery time of the failed computational node.

Proactive mechanisms, on the other hand, are able to predict a potential fault before it even occurs and to operate in a proactive manner in order to minimize the effects of the fault on the infrastructure [6]. This is possible via the use of dedicated prediction mechanisms. The prediction mechanisms consume data that are collected during the various operational stages. The appropriate response actions are then triggered as a response. The proactive approaches leverage methodologies such as migration [7] and load balancing [8]. Preemptive migration relies on the prediction of faults in order to relocate the tasks that may be potentially affected to another computational node. In proactive replication methodologies, a specific node is replicated in a proactive manner based on a prediction that is provided by a dedicated mechanism. The term computational node in the context of this paper refers to any type of entity with processing capabilities. Thus, this term encompasses Virtual Machines (VMs), containers, PODs in Kubernetes or even physical devices.

The strain that is caused in infrastructures due to sudden increases in task production is integrally intertwined with the traffic that traverses this infrastructure and the number of service requests. Thus, having access to predictions regarding them is extremely useful in order to implement optimal resource management strategies of the infrastructure[9] and supporting functionalities such as load balancing and resource allocation in order to fulfill the QoS requirements. Furthermore, the emergence of data centers has dictated the necessity to develop more suitable methods to deal with the complex properties of high-dimensionality and long-range dependencies [10].

Point process statistical models like Poisson processes were initially used but they soon presented the drawback that they are not able to properly encapsulate the self-similarity characteristics [11] that are inherit in sequence values. Another approach that was later developed involved the use of time series models such as Autoregressive–Moving-Average (ARMA) and their variations Autoregressive Integrated Moving Average (ARIMA) and Seasonal ARIMA (SARIMA) [12].

Multi-step prediction methodologies are especially useful in cases that include short and long range time dependencies [13]. Multi-step prediction using Recurrent Neural Networks has been applied for IoT traffic time series prediction [14]. This approach is based on the assumption that for each step prediction the output of the Recurrent Neural Network is used as input in order to create the prediction for the next time-step. The drawback of this approach is that such feedback loops are not directly designed for sequence prediction and thus tend to accumulates errors over numerous steps.

Sequence to sequence (seq2seq) architectures are able to optimally encapsulate the underlying temporal correlations and produce predictions that correspond to various time steps. A notable approach for seq2seq modeling is the encoder-decoder [15] that consists of one neural network that maps the input sequence of previous steps to an intermediate vector and the decoder which maps the intermediate vector to a sequence prediction. Encoder-decoders have been used in multiple fields for multi-step prediction but they have not been used in service traffic prediction. Especially the multi-step predictions in the domains of transportation [16] and spatio-temporal mobility [17] present numerous similarities in regards to their perspective problem formulation and data structure with the prediction of the number of tasks being produced. An encoder-decoder that leverages a Convolution Neural Network encoder and a Recurrent Neural Network decoder has been used in order to conduct intelligent transportation planning [18].

Our work aims to introduce a number of mechanisms that are able to leverage the benefits that are provided by the multi-step format in a more refined manner. Having access to information regarding multiple future instances allows us to design automated resource orchestration strategies that cater to the specific characteristics of each type of computational node.

3 Proactive Fault Tolerance

Edge Computing paradigms heavily rely on a vast number of computational nodes that operate simultaneously. Thus, it is of paramount importance to consider failures as a statistical inevitability. To that end, it is essential to guarantee that the Edge Infrastructure will be able continue operating without interruptions and QoS deterioration in the event of component failure.

The replication process of a node, such as a VM, requires a certain waiting period, which would provoke QoS degradation. Thus, Fault Tolerance should be achieved by following a proactive approach. At any given time, the network should contain a specific number of computational nodes, which can remain idle until one of the already operating components ceases to function properly. Given that redundant computational nodes may be requested, it is important to keep this redundancy to a minimum. However, by utilizing machine learning algorithms, it is possible to extract information regarding the behaviour patterns of the services and the process faults that occur. Hence, this enables the Fault Tolerance functionality to manifest in a manner which will ensure that the operations will continue to take place uninterrupted.

The proposed APAFT incorporates fault predictions by leveraging information that is associated with task production rates in the Edge Infrastructures and the subsequent ability of the Infrastructures to handle these tasks. The APAFT is able to closely monitor the number of tasks that are sent to each computational node to reveal potential bottlenecks in task execution that may result in potential QoS degradation. Furthermore, the APAFT is designed to leverage information that is specific to each type of computational node. In the event that the deployed resources cannot satisfy the increasing amount of demands that is expected to occur within a specified time-frame, the APAFT will then trigger proactive node replication.

3.1 System Architecture

The APAFT architecture consists of six main components as depicted in Figure 2.

- The Storage Component that contains information regarding each type of computational node included in a specific Edge Infrastructure. This includes the time required for the replication process of the computational node to take place as well as the maximum number of tasks per minute the computational node can handle before showing any signs of performance degradation.
- The Monitoring Agent that is in charge of monitoring the number of tasks that are sent to each computational node.
- The Prediction Mechanism that is responsible for performing accurate multi-step predictions in regards to the number of tasks that shall be sent to each computational node. It is essential for the Prediction Mechanism to produce a multi-step output. Deep Learning methodologies have shown great promise in terms of sequence-to-sequence modeling and even surpass many well-established alternatives. As a result, we decided to use our novel DL-based Encoder-Decoder mechanism that provided excellent results in terms of predicting network traffic [19].
- The Matching Mechanism that receives as input the multi-step prediction generated by the prediction mechanism and the time that is required to replicate this particular type of computational node. The latter input is provided by the Storage Component. Upon receiving these two inputs, the Matching Mechanism shall decide which one of the steps that belong to the multi-step prediction should be kept in order to produce a single-step prediction. The decision process is based on the fact that it is essential to choose a time-step that is farther into the future compared to the moment that the replication process shall be completed. On top of that, it is important to choose the time-step that is closer to the moment that the replication process is expected to be over since the farther one tries to look into the future the less accurate the predictions will be. This process is able provide predictions that are specific to each type of computational node. An example of how the Matching Mechanism decides which prediction step to keep is depicted in Figure 1.
- The Stabilizing Mechanism that is in charge of ensuring that the fluctuations inherited in task production rate will not affect negatively the overall efficiency of the suggested pipeline. In case that a prediction about a specific computational node, is describing a substantial drop in task production, then that particular node shall be decommissioned in order to avoid resource over-provisioning. This drop may not be indicative of an overall trend in task production, but rather of a singular drop. If that is indeed the case, then the ramifications on the pipeline's ability to provide advanced fault-tolerance shall be dire since there will be a shortage of resources. In order to avoid this scenario, a Stabilizing Mechanism is introduced. The Stabilizing Mechanism takes into consideration the ongoing task production, as well as the single-step prediction which is produced by the Matching Mechanism, in order to produce a hybrid task production metric that is more tolerant to the fluctuations in task production.
- The Scaling Mechanism receives as input the single-step prediction that is produced by the Matching Mechanism and the maximum number of tasks per time-step that this particular type of computational node can handle. The latter input is provided by the Storage Component. Upon receiving these inputs, the Scaling Mechanism is responsible for deciding whether this particular computational node should be replicated in case the prediction is greater than the maximum number of tasks per minute the computational node can handle, or

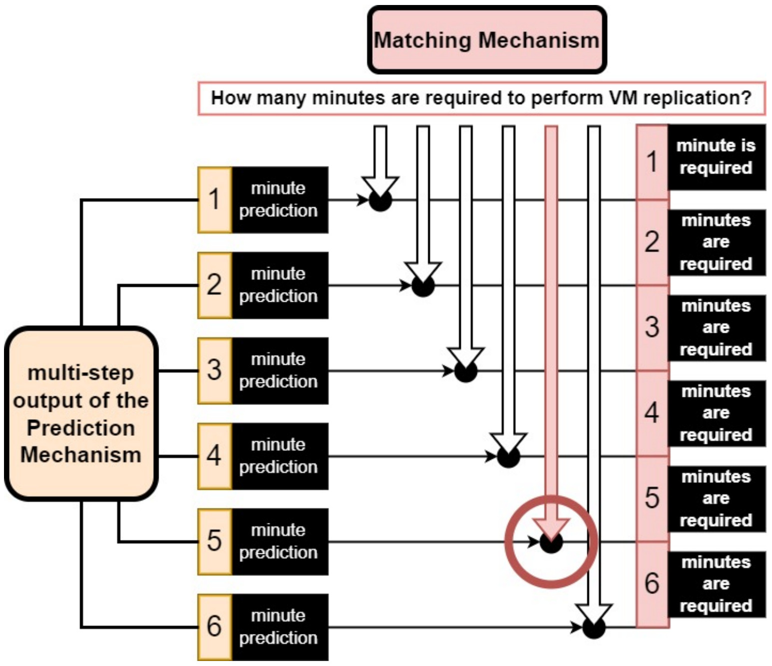

Figure 1: The Matching Mechanism maps the time that is required to deploy a specific VM to the appropriate step of the prediction.

de-allocated in case the prediction is lower than a specified threshold. This threshold is used in order to avoid over-provisioning of resources.

4 Experimental Evaluation

4.1 Simulation

In order to examine the efficiency of the proposed automated pipeline, a large-scale simulation that lasted 1 week and included over 1 million tasks was conducted. The simulation was created using the CloudSim Plus framework[2]. Every second numerous tasks are created. The task production rate is modeled after 5 Gaussian probability distributions, in order to establish realistic traffic scenarios. Two types of task creation patterns are considered. The first one is introduced in order to simulate the fluctuation of tasks that is expected to take place during each day and the second one is used in order to create numerous sudden bursts in task production. In the context of the experiments, the Edge Infrastructure consists of 10 Virtual Machines. Half of them require 5 minutes while the rest of them require 3 minutes to be replicated. Furthermore, the former can process at most 50 tasks per minute, while the latter can handle at most 30 tasks per minute. The information regarding the number of tasks each type of VM can handle as well the time that is required, are stored in the Storage component. Once every minute the decision making process in regards to scaling takes place. There are two distinct scenarios that were examined in order to support our claims. The first one is based on a standard approach that leverages the information that is stored in the Storage component and the ongoing task production rate, in order to formulate decisions about scaling up or down. The second one fully incorporates the APAFT paradigm that is proposed by the authors of this paper.

4.2 Evaluation Metrics

In order to evaluate the efficiency of the proposed APAFT architecture, several fault tolerance evaluation metrics [20] were utilized. Mean Time To Repair (MTTR) is defined as the time that is required for the repair functionalities to take place after a failure occurs. These repair processes take place for each computational node in an independent manner. Mean Time To Failure (MTTF) corresponds to the expected time to failure given that the overall infrastructure operates in a normal manner. Within the context of this paper, this is equivalent to the average time until an end-to-end delay that

[2]`https://cloudsimplus.org/`

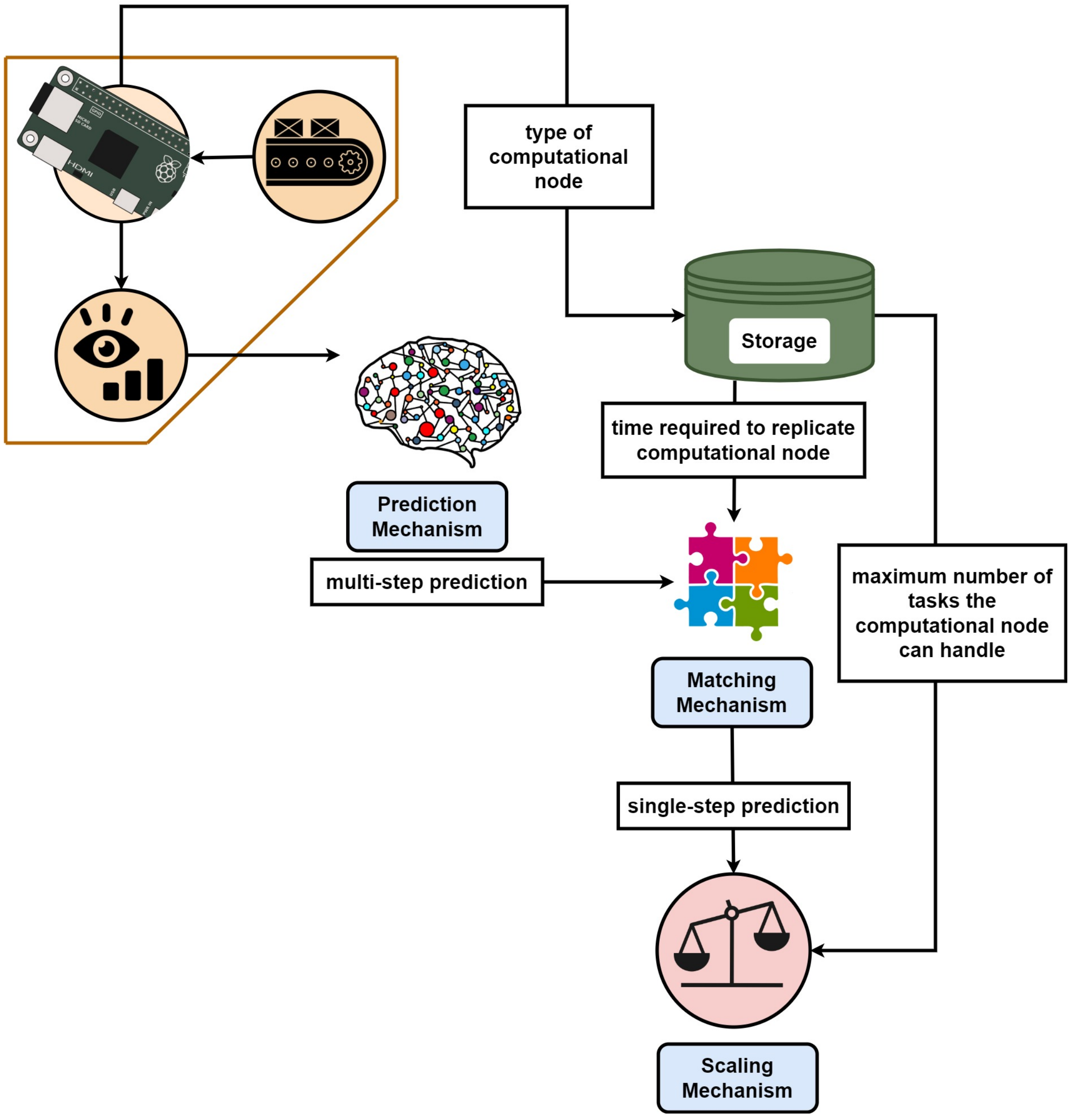
type of computational node
Storage
time required to replicate computational node
Prediction Mechanism
multi-step prediction
maximum number of tasks the computational node can handle
Matching Mechanism
single-step prediction
Scaling Mechanism

Figure 2: The architecture of the Autonomous Pipeline for Advanced Fault Tolerance (APAFT).

Table 1: Evaluation of the proposed APAFT method

	Reactive	APAFT
MTTF	6.007	13.912
MTTR	32.825	3.352
Reliability	0.857	0.932
Maintainability	0.029	0.229

surpasses the 1 second threshold is recorded. MTTF is an evaluation metric that corresponds to the overall inability of the Edge infrastructure to operate properly. It is calculated by examining the faults that take place regardless of the actual computational node that failed. Both these evaluation metrics are calculated in terms of seconds.

Reliability and Maintainability are two additional Fault Tolerance evaluation metrics are the. Reliability refers to the ability of an Edge infrastructure to operate continuously without presenting any type of failure. Maintainability refers to how easily a failed system can be repaired. Both Reliability and Maintainability are numbers with no units and higher values mean better performance.

4.3 Evaluation Results

Table 1 demonstrates the experimental results.

Upon the completion of the aforementioned experiments, the results demonstrated in Table 1 were obtained. The results are the following ones:

- The MTTF was increased in the scenario that the APAFT was incorporated in the decision making process.
- The MTTR was significantly reduced in the scenario that the APAFT was incorporated in the decision making process.
- The Reliability was improved in the scenario that the APAFT was incorporated in the decision making process.
- The Maintainability was significantly improved in the scenario that the APAFT was incorporated in the decision making process.

These results highlight the efficiency of the proposed pipeline. The APAFT contributes towards establishing Edge Computing paradigms that are less prone to manifesting faults and more capable at recovering.

5 Conclusion

The rather complex, dynamic and heterogeneous nature of the Edge Infrastructures pose significant challenges in establishing efficient Fault Tolerance functionalities. In this paper, we proposed a automated pipeline that aims at providing proactive fault tolerance capabilities by leveraging multi-step predictions and numerous components implemented in a way that takes into consideration the intricacies inherited in Edge Infrastructures. In addition, these components have been designed to fully exploit the richness of information that is provided by the multi-step prediction format. This pipeline consists of six main components, Storage Component, Monitoring Agent, Prediction Mechanism, Matching Mechanism, Stabilizing Mechanism and Scaling Mechanism. The effectiveness of the proposed architecture is evaluated in the context of a large-scale simulation using several Fault Tolerance metrics.

Acknowledgment

This work is part of the CHARITY project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016509".

References

[1] Antonios Makris, Abderrahmane Boudi, Massimo Coppola, Luís Cordeiro, Massimiliano Corsini, Patrizio Dazzi, Ferran Diego Andilla, Yago González Rozas, Manos Kamarianakis, Maria Pateraki, Thu Le Pham, Antonis Protopsaltis, Aravindh Raman, Alessandro Romussi, Luís Rosa, Elena Spatafora, Tarik Taleb, Theodoros Theodoropoulos, Konstantinos Tserpes, Enrico Zschau, and Uwe Herzog. Cloud for holography and augmented reality. In *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*, pages 118–126, 2021.

[2] Theodoros Theodoropoulos, Antonios Makris, Abderrahmane Boudi, Tarik Taleb, Uwe Herzog, Luis Rosa, Luis Cordeiro, Konstantinos Tserpes, Elena Spatafora, Alessandro Romussi, et al. Cloud-based xr services: A survey on relevant challenges and enabling technologies. *Journal of Networking and Network Applications*, 2, 2022.

[3] Rong Chen, Youyang Yao, Peng Wang, Kaiyuan Zhang, Zhaoguo Wang, Haibing Guan, Binyu Zang, and Haibo Chen. Replication-Based Fault-Tolerance for Large-Scale Graph Processing. *IEEE Transactions on Parallel and Distributed Systems*, 29(7):1621–1635, July 2018. Conference Name: IEEE Transactions on Parallel and Distributed Systems.

[4] K. Vinay and S.M. Dilip Kumar. Fault-Tolerant Scheduling for Scientific Workflows in Cloud Environments. In *2017 IEEE 7th International Advance Computing Conference (IACC)*, pages 150–155, January 2017. ISSN: 2473-3571.

[5] Pekka Karhula, Jan Janak, and Henning Schulzrinne. Checkpointing and Migration of IoT Edge Functions. In *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking*, EdgeSys '19, pages 60–65, New York, NY, USA, March 2019. Association for Computing Machinery.

[6] Yuli Tian, Jeff Tian, and Ning Li. Cloud reliability and efficiency improvement via failure risk based proactive actions. *Journal of Systems and Software*, 163:110524, May 2020.

[7] Alexander Power and Gerald Kotonya. A Microservices Architecture for Reactive and Proactive Fault Tolerance in IoT Systems. In *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pages 588–599, June 2018.

[8] Priti Kumari and Parmeet Kaur. A survey of fault tolerance in cloud computing. *Journal of King Saud University - Computer and Information Sciences*, 33(10):1159–1176, December 2021.

[9] M. Adel Serhani, Hadeel T. El-Kassabi, Khaled Shuaib, Alramzana N. Navaz, Boualem Benatallah, and Amine Beheshti. Self-adapting cloud services orchestration for fulfilling intensive sensory data-driven IoT workflows. *Future Generation Computer Systems*, 108:583–597, July 2020.

[10] Xiaofeng Cao, Yuhua Zhong, Yun Zhou, Jiang Wang, Cheng Zhu, and Weiming Zhang. Interactive Temporal Recurrent Convolution Network for Traffic Prediction in Data Centers. *IEEE Access*, 6:5276–5289, 2018. Conference Name: IEEE Access.

[11] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995. Conference Name: IEEE/ACM Transactions on Networking.

[12] V. Eramo, T. Catena, F.G. Lavacca, and F. di Giorgio. Study and Investigation of SARIMA-based Traffic Prediction Models for the Resource Allocation in NFV networks with Elastic Optical Interconnection. In *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, July 2020. ISSN: 2161-2064.

[13] Filip Pilka and Miloš Oravec. Multi-step ahead prediction using neural networks. In *Proceedings ELMAR-2011*, pages 269–272, September 2011. ISSN: 1334-2630.

[14] Ali R. Abdellah, Omar Abdul Kareem Mahmood, Alexander Paramonov, and Andrey Koucheryavy. IoT traffic prediction using multi-step ahead prediction with neural network. In *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–4, October 2019. ISSN: 2157-023X.

[15] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678, June 2018. ISSN: 1931-0587.

[16] Zhengchao Zhang, Meng Li, Xi Lin, Yinhai Wang, and Fang He. Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies. *Transportation Research Part C: Emerging Technologies*, 105:297–322, August 2019.

[17] Xinglei Wang, Xuefeng Guan, Jun Cao, Na Zhang, and Huayi Wu. Forecast network-wide traffic states for multiple steps ahead: A deep learning approach considering dynamic non-local spatial correlation and non-stationary temporal dependency. *Transportation Research Part C: Emerging Technologies*, 119:102763, October 2020.

[18] Zhumei Wang, Xing Su, and Zhiming Ding. Long-Term Traffic Prediction Based on LSTM Encoder-Decoder Architecture. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2020. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[19] Theodoros Theodoropoulos. An-Encoder-Decoder-Deep-Learning-Approach-for-Multistep-Service-Traffic-Prediction, May 2021.

[20] Salma M. A. Ataallah, Salwa M. Nassar, and Elsayed E. Hemayed. Fault tolerance in cloud computing - survey. In *2015 11th International Computer Engineering Conference (ICENCO)*, pages 241–245, December 2015.