Network Slice Monitoring System For Disaster Detection

Rodrigo Sanches Miani[1], Regina Melo Silveira[2], Rafael Pasquini[1] , Luciana Arantes[3] and Pierre Sens[3]

[1] Universidade Federal de Uberlândia, Minas Gerais, Brazil

[2] Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, trav. 3, no. 158, São Paulo, Brazil

[3] Sorbonne Université, CNRS, LIP6, Paris, France

{miani,rafael.pasquini}@ufu.br, regina.silveira@usp.br,{luciana.arantes, pierre.sens}@lip6.fr

Abstract. *The use of monitoring systems for risk areas has increased significantly, especially with the use of IoT applications, which has driven Early Warning Systems (EWS). However, risk situations due to natural disasters can cause malfunction of communication networks making such monitoring unfeasible. In this paper we propose an unorthodox method for disaster detection, based on sliced network traffic monitoring. In the presented system the network traffic is monitored by a Monitoring Agent module that uses a Random Forests method to identify anomalous network traffic events. Simulation results show that our proposal is a promising solution for disaster detection, with the advantage that it could be used in different situations.*

1. Introduction

Every year, several regions of the planet are hit by natural disasters such as earthquakes, hurricanes, tsunamis, seismic tremors, floods, and landslides, causing extensive human and economic damage. The current communication networks technologies and the Internet of Things (IoT) have enhanced and boosted numerous applications, including those for monitoring these events, with great benefit to society [Porte et al., 2020]. However, several challenges must still be overcome to ensure that these disasters could be detected quickly and despite network damage, which motivate this work.

The problems caused by natural disasters have become so serious that the United Nations Office for Disaster Risk Reduction (UNDRR)[1] has proposed a Sendai framework for disaster reduction that recommends substantially increasing availability and access to multi-hazard warning systems by 2030 [Sendai, 2015]. Due to these issues, a new research field is emerging called Early Warning [Esposito et al., 2022]. The high capacity of data mining and processing, arising from Big Data mechanisms, and the ability to automate processes, using Artificial Intelligence (AI) and Machine Learning (ML), make Early Warning System (EWS) possible.

Despite the various systems proposed and deployed in a traditional EWS, the time required to collect data, send it to the central station, and process it is still a system bottleneck. As highlighted by the authors in [Esposito et al., 2022] the data analysis latency is the main component in this weakness. Reducing this time is of utmost importance, justifying our work, mainly because the faster an alarm is

[1] https://www.undrr.org/

triggered or rescue information is sent, the higher the number of saved lives and the lower the damage.

The communication lack due to disasters make it difficult, or even impossible, to disclose the event, and consequently the activation of rescue actions. Therefore, the need to design and implement mechanisms to detect network failures has proven to be paramount for risk situations, in natural disasters or even in provoked events that endanger human life or the environment [Gomes et al., 2016].

An unorthodox way to make early disaster prediction is from the communication network operation monitoring. In the occurrence of some event that causes communication signal deterioration, such as wireless network signal degradation by unexpected interference, network equipment breakdown or partial or total optical fiber rupture, the network performance can be used as an early identification parameter for disaster prediction.

The projects STIC AmSud/CAPES ADMITS and ITERATION-D aim to create data analysis methods to verify disaster situations. In order to reach this goal, an architecture based on Sliced Network (SN) [Afolabi et al., 2018] [Foukas et al., 2017] is considered, using the Software Defined Networks (SDN) [Kreutz et al., 2014] and Virtualized Network Functions (NFV) [Mijumbi et al., 2015] paradigms. In this paper we propose the inclusion of a monitoring agent that can be strategically installed in the orchestration element monitor that verifies the risk detection application`s network flow to infer a disaster occurrence in advance.

The remainder of this paper is organized as follows. Section 2 brings related works, while the problem characterization is presented in Section 3. Section 4 describes network architecture considered, and discusses its requirements and issues that arise in the considered scenario. Section 5 describes the slice monitor agent and the machine learning method that are used. The results of method simulation and performance analysis of the proposed solution are presented in Sections 6. Finally, Section 7 presents our concluding remarks and considerations for future work.

2. Related Work

Due to its relevance, the subject of natural disasters has been recurrently found in the scientific literature. Our present paper focused on the identification of disaster types, how to detect the event, the use of network technologies in systems to predict and detect events in risk areas, and the impact on them in case of disaster.

Under Early Warning, Esposito et al. (2022) survey how IoT infrastructure can be used on this approach using four case studies: tsunami, flood, earthquake, and landslide. The work is quite extensive and complete, considering IoT applications, such as monitoring and surveillance, highlighting the services' vulnerabilities in terms of recovery and resource restriction. Also, the work of Linardos et al (2022) presents a review of scientific contributions from 2017 to current days focusing on predicting disasters, risk detection and vulnerability assessment by EWS using ML and deep learning (DL).

Scardino et al. (2022) used coastal video monitoring coupled with machine learning and computer vision techniques to detect events in the sea. They use Convolutional Neural Network (CNN) to assess tide, wave, and storm parameters from a video record that can predict hurricane formation on the coast.

Recent natural disasters have revealed that the network infrastructure is

currently very susceptible in the face of such events, making it difficult to deploy and coordinate relief operations. Thus, the concern about communications network damage in hazardous areas have been widely discussed in the literature, due to the high impact on information dissemination, such as alerts and rescue actions [Gomes et al., 2016].

Since disaster effects do not exhibit deterministic behavior, disrupting specific parts of the network could happen. Communication infrastructure vulnerabilities can be mapped [Agarwal et al., 2013] and strategies are proposed for network resilience through routing mechanisms [Jahir et al., 2019], redundant architecture, integration of network technologies using High Frequency (HF) communications with ionospheric reflection, satellite communications, and mini-HAPs (High Altitude Platforms) [Porte et al., 2020] and recovery techniques to ensure that communication in the risk area vicinity is not affected by the event, mainly because network reestablishment can take weeks.

Besides, the RECODIS project [Esposito et al., 2018] addressed the problem to protect the network against disasters and to ensure service continuity considering 5G technology, and claiming that softwarization features, such as NFV and SDN, improve the network recovery conditions at runtime. Gomes [Gomes et al., 2016] discusses the issues of network disruption vulnerability due to disasters, proposing a routing algorithm resilient to these events.

Some research projects have explored the issue of fault detection in a communication network with different goals and purposes. Considering distributed systems, Rossetto [Rossetto et al., 2018] proposed an untrusted fault detector, called Impact FD, where a node monitors a set of nodes (sensors, devices, processes), and the monitoring node, FD oracle, sends its trust to the set of monitored nodes as a whole and not for each of these nodes, to ensure a fault-free connection.

The work of Khan, Gupta and Gupta [Khan, Gupta and Gupta, 2020] describes in detail technologies used by disaster detection and events consequences on network infrastructure. Their survey focuses on solutions for early detection, prevention, recovery, and management of risk situations, considering application and network troubleshooting and maintenance as a part of system management.

The present work does not aim to discuss network recovery, but use the network signal anomaly facing a disaster to identify the event occurrence. Although IoT applications and EWS have been explored and described by many authors to risk areas monitoring, according to the authors' knowledge, this is the first paper that addresses network monitoring to disaster detection. Besides, our proposal has the distinction of being of general scope, regardless of the occurrence of disaster type, and it can anticipate the event before the environmental monitoring signal has been processed at the central station.

3. Problem Characterization

This work considers a traditional monitoring service, based on an IoT application, where the risk environment is monitored. The signal generated is sent, via a SN, to a monitoring center station, where the data is processed. The monitoring center station must be geographically distant from the risk area to reduce the probability of being hit by the disaster [Agarwal et al., 2013].

The main idea is that the traffic flow coming from the IoT application is

monitored to infer, in advance, if there has been a disaster. This inference is based on the network signal variation and anomalies, comparing the current traffic with historical ones. As Afolabi et al. [Afolabi et al., 2018] described in their survey, SN must ensure the principles related to the network operation, that are automation, isolation, customization, elasticity, programmability, end-to-end, and hierarchical abstraction. The isolation, key characteristic for the network resources sharing, can be reached by Network Operational System (NOS) instantiation. Hence, the slice will have resources guaranteed in an exclusive way, which ensures no competition with others flow connections.

Since the data flow is transmitted by a dedicated slice, it will not suffer interference from other transmission. The network should have a recovery mechanism, such that a network failure will be detected in an average time of 65 ms (3GPP). Thus, any interference or signal change detected can be identified as a network external unexpected event.

Natural disasters can be of different types, such as hurricanes, tsunamis, floods or earthquakes. As described by Esposito et al. [Esposito et al., 2022], each of them has different characteristics, but most of them can cause network disruptions that could negatively impact the communication infrastructure.

Considering the main types of natural disasters, they can be monitored using different methods, either from motion (accelerometers), vibration (seismometers) or temperature sensors, or from cameras, either thermal or normal ones, or from images. Such methods are based on capturing and processing the information, frequently using machine learning to compare with expected data, after a period of machine training with historical data [Esposito et al., 2022][Im and Takeuchi, 2019][Furquim et al., 2018][Linardos et al., 2022].

The monitored data has different characteristics and traffic profiles. Sensor data typically present burst rate (BR), for periodic measurements, or constant bit rate (CBR) traffic profiles, while camera monitoring produces compressed images with MPEG or similar standards, which results in a variable bit rate (VBR) traffic profile. Table 1 correlates the disaster events considered in this work, the detection methods frequently used for specific EWS, and traffic characteristics produced for each detection device.

As mentioned before, disasters are non-deterministic events, thus all network elements are subject to be damaged. In addition, because of changes in the monitored environment, several types of interference can modify data measurements. The network anomalies can be verified by its performance using parameters such as Packet Delivery Ratio (PDR), Routing Overhead, End to End Delay (E2E delay), Packet Loss Ratio (PLR), average energy consumption, and average throughput [Khan, Gupta and Gupta, 2020]. By the network traffic approach, a disaster can cause different consequences in the risk detection IoT application data flow, depending on its severity and the degree of damage caused. In the list below we have identified some possible damage consequences:

- Change in the signal profile, increased due to the change of parameters detected;
- Change in the signal profile, increased due to interference;
- Change in the signal profile, decreased due to network degradation based on external interference, including large delay and low rate packet loss;
- Loss of the signal, due to total network disruption.

Thus, we argue that, considering that IoT device connection to the central station is based on a sliced network, the traffic profile can carry relevant information regarding

the monitored environment. Hence, the network traffic profile information will be used as a parameter to infer a disaster event in a hazardous area.

Table 1. Characterization of the natural disasters types, correlated monitoring methods, and the generated traffic profile

Disaster Event	Geographic coverage	Monitoring Mechanism	Network Traffic Characteristic
Hurricanes	Regional	Wind waves measurement	CBR
		Video surveillance	VBR
Tsunamis	Regional	Water level measurement	BR
		Wind waves measurement	CBR
		Video surveillance	VBR
Floods	Local	Water level measurement	BR
		Video surveillance	VBR
Earthquakes	Regional	Seismometer	BR
		Accelerometer	BR
		Video surveillance	VBR
Landslide	Local	Accelerometer	BR
		Video surveillance	VBR

4. Network Environment

Currently, several network technologies foresee the use of network slicing. Such technique is considered the key to serve millions of competing connections, which use the same network infrastructure, without generating competition among them. This happens because slicing allows network instance creation, with exclusive resource allocation, with guaranteed traffic isolation and flexibility to meet end-to-end connections with different requirements [Afolabi et al., 2018].

The network slicing brings as main benefits isolation and flexibility, and can be deployed end-to-end or in specific network segments [Afolabi et al., 2018]. The end-to-end model enables the slicing configuration at the network entrance, setting it up according to the contracted QoS and SLA requirements. The slicing can be offered by the network infrastructure provider (NP - Network Provider) as a service, NSAAS (Network Slice As A Service) [Clayman et al, 2021], and can be managed by the NP itself or by the service provider (ISP - Internet Service Provider).

This project is based on the NECOS architecture [Farias et al, 2019], which presents the following advantages: i) a very generalist architecture, therefore allowing its coupling to several network technologies, such as 5G, IEEE 802.16, or even optical technologies such as WDM; ii) a softwarized architecture, which allows integration with SDN functions; iii) end-to-end resource reservation guarantee based on network slice and NFV; and iv) a distributed architecture, where Slice Agents installed in different network sectors (edge, network, and core), and in different

domains, participate in the slices administration and orchestration, to ensure good end-to-end performance.

Therefore, considering the NECOS architecture, we have a Slice Orchestrator and the IMA (Infrastructure & Monitoring Abstraction) that through the northbound interface monitors the slice lifecycle. Figure 1 illustrates the NECOS architecture and its modules.

In principle, the sliced network, implemented using network instantiation, guarantees resources by reserving them in advance, before the beginning of data sending. Thus, any network disturbance that does not meet the specified QoS nor SLA certainly does not occur due to competition for resources, as is the case in non-sliced networks [Yu et al., 2020]. Hence, it is possible to infer that such disturbance is caused by an event external to the network, which may characterize a disaster.

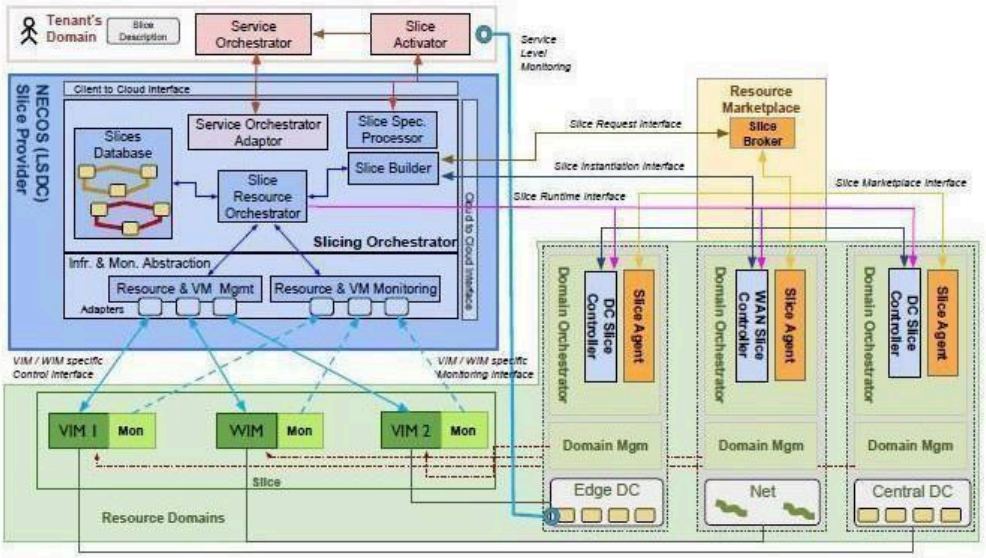

Figure 1. NECOS Architecture [Farias et al., 2019]

5. Slice and Monitor Agents

For performing an end-to-end network monitoring, with identification of failure cases due to disasters, it is necessary to elaborate a distributed architecture with active monitoring. For this, a monitoring agent must be active at specific points, in the orchestrating element of each domain, as shown in Figure 2.

The distributed architecture proposed in this work will not serve all connections, but only the connections with specific requirements to support risk area monitoring applications. Therefore, based on the SLA criteria there is a network state in which this monitoring service is requested from the service negotiation.

As mentioned before, this work considers the use of a network based on slicing, since the risk area monitoring service connection uses an exclusive slice, defined through the previously established SLA contract. In this way, all traffic coming from the monitoring service can be processed by the Monitor Agent.

The monitoring agent must use real-time data analysis algorithms that must be designed to perform efficiently, outperforming, in terms of time, IoT devices for monitoring risk areas. To this extent, it is possible to obtain quick response in critical

situations and trigger alert signals in case of disaster. Techniques involving learning processes must be used to improve agents' performance, making them adaptive and less sensitive to false positives.

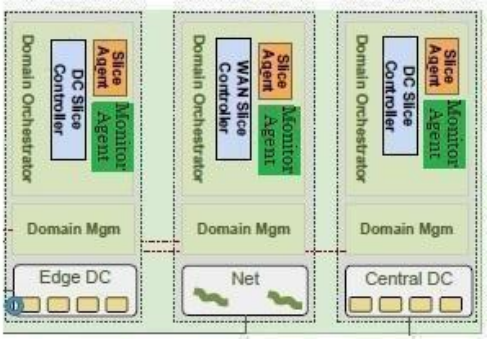

Figure 2. Partial View of the NECOS Architecture showing the inclusion of the Monitor Agent (Modified from [Farias et al., 2019])

5.1. Automation Mechanisms

Network anomalies can be detected through the analysis of the network behavior and transmission service degradation. The mechanism must be reactive, that is, as soon as an anomaly is detected, an action must be taken immediately. The shorter the time between detection and action, the greater the chances of preventing and reversing the situation. Therefore, this detection mechanism should be performed automatically.

Several mechanisms have been proposed in the literature for the automation of network anomaly detection. They use network parameters, such as delay or packet loss, to verify the network behavior. As described by Nassif et al. (2021), machine learning (ML) has proven to be efficient in detecting anomalies and network behavior failures. Among the possibilities of using ML for fault identification is the Classification method, which can be implemented in the form of supervised learning [Boutaba et al., 2018], using QoS parameters as target values.

As Kwon et al. (2017) describe, other ML methods also used for network fault detection are Deep Learning (DL) methods, considering the following algorithms: Restricted Boltzmann Machine (RBM), Deep Boltzmann Machine (DBM), Deep Neural Network (DNN) and Recurrent Neural Network (RNN).

Nevertheless, Random Forest (RF) has been used for disaster detection in risk areas as described by Im et al. (2019) and Linardos et al. (2022), presenting good performance. Proposed by L. Breiman in 2001 [Breiman, 2001], it also demonstrates to be adequate to classify network anomalies based on the historical data [Boutaba et al., 2018]. According to Geurts et al. (2006), the RF algorithm is based on constructing multiple decision trees, each operating as a classifier. Each tree is sampled from the original dataset to build a sub-dataset. The sub-datasets are put into each decision tree, and each decision tree outputs a result. The voting of all decision trees determines the final decision result. The RF algorithm can be used in both classification and regression tasks. This work will use the RF regression algorithm to identify network anomalies by evaluating network traffic statistics.

5.2. Monitoring Agent Operation

Since our scope is developing a system to anticipate issues in disaster monitoring infrastructure, it is important to describe how the monitoring agent described in Figure 2 would act in such a scenario. Figure 4 illustrates a use case for our monitoring agent.

The environmental sensors send periodic data to the Monitoring center along a path with multiple network switches. The Monitor Agent of each switch reads the input data sent by the sensors and uses a previously trained RF model to predict what the last switch C will forward to the Monitoring center. The Monitor Agent of switch A sends to the Monitor Agent of switch C, the closest one to the Manager Center, the prediction it made. The latter can thus compare the actual values it received with such a prediction. If the current value is different from the predicted value (outside a predefined range, for example) then an alert is sent to the Monitoring center as depicted in Figure 4.

6. Results and Analysis

In order to validate our proposal, we conducted several experiments using a public dataset. Our idea is to evaluate the scenario described in Figure 4 using network traces (KTH Dataset) from a testbed composed of several machines that simulate a scenario running two services: video-on-demand and distributed data storage [Stadler, Pasquini and Fodor, 2017]. Section 6.1 details the network trace. Python language is used with scikit-learn to build the RF model and perform the experimental validation.

The KTH dataset consists of network traces collected from a testbed deployed at the KTH - Royal Institute of Technology in Stockholm. The testbed includes a server cluster, an emulated OpenFlow network, and a set of clients. Figure 3 illustrates the testbed. The main goal of the testbed is to predict end-to-end service-level metrics from low-level infrastructure measurements.

The network trace is composed of two types of measurements: Infrastructure statistics - here we have two feature sets: i) cluster statistics (CPU utilization, memory utilization, disk I/O and so on) and ii) network statistics collected from the OpenFlow switches per port granularity level (number of Bytes Transmitted per port, number of Bytes Received per port, number of Packets Transmitted per port, and number of Packets Received per port.); and end-to-end service-level metrics values - i) display frame rate and audio buffer rate for the Video on-demand application and ii) read response time and write response time for the distributed data store application.

The authors proposed two network traffic patterns for each application (Video-on-demand and Distributed data store application), periodic and flash-crowd. The periodic-load pattern consists of a Poisson process whose arrival rate is modulated by a sinusoidal function with a period of 60 min. The flash-crowd-load pattern consists of a Poisson process whose arrival rate is controlled by the flash-crowd model described in [Yao et al., 2012]. During the experiments, both infrastructure and end-to-end service-level metrics values were collected every second. In this work, we will use only infrastructure statistics, particularly the network statistics collected from the OpenFlow switches.

6.1. Evaluation Metrics

Our goal is to estimate a particular network traffic measurement Y_t received at time t on the network switch closest to the Monitoring center, based on knowing the network statistics X_t in a network switch most close to the environmental sensors. Our problem then is to find a model $M : X_t \rightarrow \hat{Y}_t$, such that $\hat{Y}_t$ closely approximates Y_t for a given X_t.

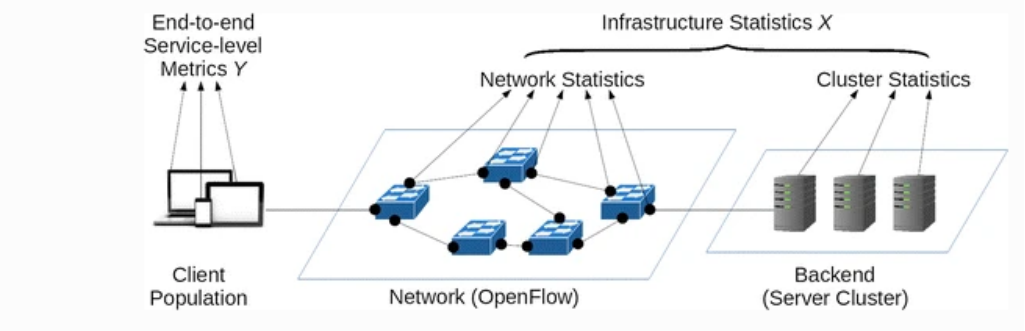

Figure 3. Diagram representing Testbed used for dataset capture (from: Stadler et al., 2017)

We evaluate the regression models using a metric known as Normalized Mean Absolute Error (NMAE). Equation 1 defines the NMAE metric:

$$NMAE = \frac{1}{\bar{y}}\left(\frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}|\right)$$

where y is the network traffic metric value received on the switch closest to the Monitoring Center, $\hat{y}$ is the model estimation for the measured metric y, $\bar{y}$ is the sample mean, and m is the test set size. Lower NMAE values indicate better performance for the regression model.

6.2. Experimental Setup

Figure 5 illustrates the testbed deployed in the KTH dataset, and how it relates to our scenario (fig. 4). The Server Cluster emulates the data sent by our Environmental sensors, and the Client emulates our monitoring center receiving data from the sensor. The network statistics is collected from switches SWB1 and SWB3. In SWB1, the transmitted data are gathered from the Environmental sensors/Server Cluster, i.e., the number of Bytes transmitted per port, number of Bytes received per port, number of Packets transmitted per port, and number of Packets received per port. This data forms our X_t set. In SWB3, a single received network metric statistic (number of Bytes or number of Packets received) are collected in each of its ports (there are two ports connected to the Client). The data forms our Y_t set. Considering that we have two ports in SWB3, two types of network metrics, and two types of network traffic (periodic and flashcrowd), the scenarios of our experiments are summarized in Table 2.

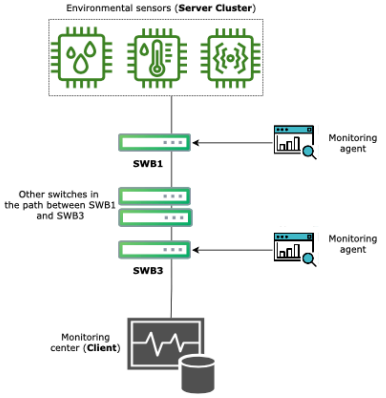

Figure 4. Monitoring Agent use case

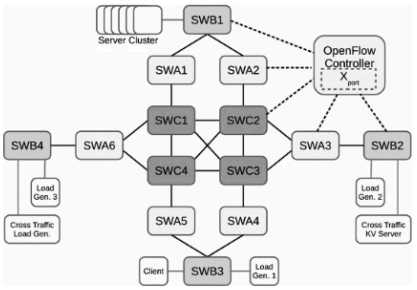

Figure 5: The KTH dataset testbed

Table 2: Scenarios summarized for experimental evaluation

Scenario	Switch Port	Metric	Network Traffic
1	A	Bytes	Flashcrowd
2	B	Bytes	Flashcrowd
3	A	Packets	Flashcrowd
4	B	Packets	Flashcrowd
5	A	Bytes	Periodic
6	B	Bytes	Periodic
7	A	Packets	Periodic
8	B	Packets	Periodic

RandomForestRegressor algorithm available on scikit-learn is used, with the n_estimators parameter equal to 10. Each scenario described in Table 2 was run 20 times.

For each run, we vary the random_state parameter of the RandomForestRegressor using a random integer between 1 to 50. Training and testing sets are divided using the hold-out strategy, with 80% of the dataset used for training and 20% for the test set. The Random_state parameter, which varies between 1 and 50, is used for each scenario to select training/test samples. The experiments run in a machine Lenovo ThinkSystem Server ST50 Xeon E-2224G 3.5Ghz 32GB.

Having small NMAE values for our proposed scenarios, we can use the transmitted data from the sensors to predict the received values in the monitoring center. This way, it is possible to identify network anomalies along the network path that might be related to a disaster. As discussed in Section 5.2, our solution uses this information to anticipate potential issues in the disaster management system.

6.3. Results

Table 3 summarizes the experimental results of our scenarios. The NMAE for each scenario corresponds to the mean NMAE values of the twenty executions. In the worst scenario, we have an average NMAE of 12.29%, and in the best scenario, we have an average NMAE of 6.95%. These results indicate that our regression model has a small error in different network traffic situations. This means that the transmitted data from the sensors can be used to predict the received values in the monitoring center. In other words, issues between sensors and the monitoring sensor can be detected with a high accuracy.

Table 3: Results summarized for each scenarios

Scenario	Switch Port	Metric	Network Traffic	NMAE (%)
1	A	Bytes	Flashcrowd	12.29
2	B	Bytes	Flashcrowd	12.17
3	A	Packets	Flashcrowd	12.01
4	B	Packets	Flashcrowd	12.09
5	A	Bytes	Periodic	7.46
6	B	Bytes	Periodic	7.15
7	A	Packets	Periodic	7.40
8	B	Packets	Periodic	6.95

The regression model evaluated on the periodic traffic type has a better performance when compared to the flashcrowd one. The nature of both network traffic types could explain this behavior. The periodic-load pattern follows a simple sinusoidal function, while the flashcrowd has several peaks distributed over time. Figure 6 illustrates the difference between both traffic types. Another interesting result is the network traffic metric received on the switch closest to the Monitoring center. Our results suggest no difference between predicting the number of bytes or packets.

Figure 7 illustrates the relationship between actual and predicted values for scenarios 1, 3, 5, and 7. In most cases, the difference between actual and expected values is minimal, corroborating the NMAE results presented in Table 3. In order to show the applicability of this result in detecting network anomalies, we investigated the likely number of anomalies that our RF model could generate during our experiments. For this, we selected a single execution in Scenario 3 to examine the number of detected anomalies according to the following Anomaly Coefficient (AC):

$$AC_i = \left| 1 - \frac{y_i}{\hat{y}_i} \right|$$

where y denotes the network traffic metric received on the switch closest to the Monitoring Center, $\hat{y}$ is the model estimation for the measured metric y, and i is a predefined time interval. If AC_i >= than a constant gamma, then an anomaly was found. In other words, if the difference between the actual measured value and the predicted one is higher than a particular value (gamma), an anomalous event might be in place. The gamma parameter can be interpreted as a tolerance threshold determined by the network operator. Lower gamma values indicate a stricter tolerance between the measured and predicted values, whereas higher values imply a more lenient threshold. As this parameter is configurable, its value may be adjusted in accordance with the operator's policies or the specific requirements of the monitored application.

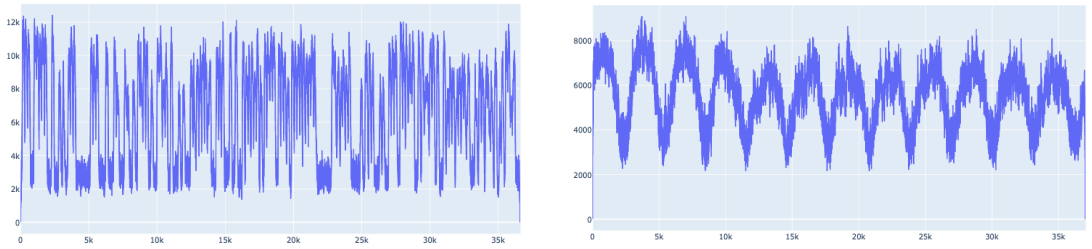

Figure 6. Results difference between both traffic types: a)periodic-load traffic and b) flashcrowd traffic.The x-axis represents the number of observations, whereas the y-axis denotes the rate of transmitted packets.

As discussed in Section 5, when an anomaly is found, we need to call a function that evaluates the current network status (CheckNetworkParameters()), and an alert is sent when the networkParameters is greater than a threshold.

As Table 4 shows (sce.3), it is possible to see that the number of detected anomalies increases inversely with the number of samples. A difference of 30% between measured and expected values led to 523 anomalies corresponding to 8% of normal samples. Table 4, for scenario 7, summarizes such a dependency. When analyzing the number of network anomalies in a scenario where the network traffic is somewhat more stable (Scenario 7 - periodic-load pattern), the number of anomalies

decreases when selecting higher gamma values.

Thus, based on the obtained results evaluation, the feasibility of establishing mechanisms for natural disasters event early detection using anomalies in the operation of the data transmission network of the IoT-based video signal monitoring service infrastructure as a verification parameter is observed. Such a mechanism can be integrated and used in various network infrastructures, since the network core is set up with slicinging network technique.

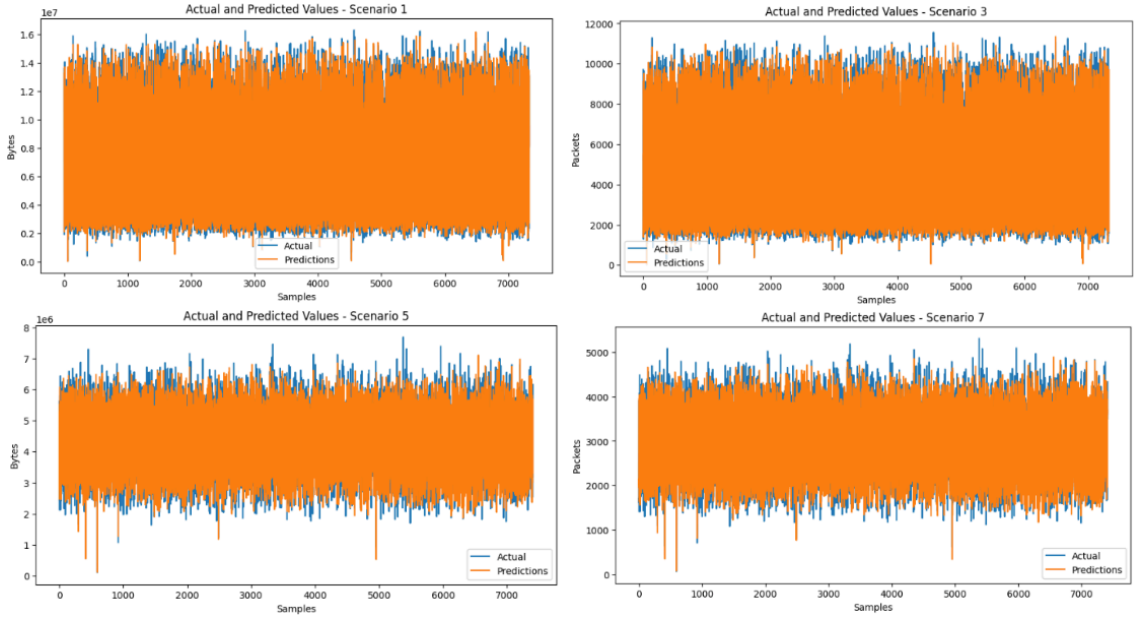

Figure 7. The relationship between the actual and predicted values for one of the 20 experimental runs conducted in scenarios 1, 3, 5, and 7.

Table 4. Variation of gamma values for a single execution of scenario 3 and 7

Scenario 3				Scenario 7			
gamma	**# of anomalies**	**# of normal samples**	**ratio**	**gamma**	**# of anomalies**	**# of normal samples**	**ratio**
0.05	2184	5143	42.47%	0.05	2269	5139	44.15%
0.10	1610	5717	28.16%	0.10	1089	6319	17.23%
0.20	918	6409	14.32%	0.20	159	7249	2.19%
0.30	523	6804	7.69%	0.30	20	7388	0.27%

7. Final Considerations

This paper proposes an architecture that uses a monitoring agent for fault detection in a communications network with the goal of predicting the occurrence of a disaster in a hazardous area. The proposal is based on the fact that checking for network anomalies is faster than the monitoring or surveillance of IoT systems. This proposal is only feasible with the use of network slicing techniques, where resources are secured in isolation and without competition with other connections. The proposal uses the NECOS platform architecture as a basis, which provides network slicing and the possibility of integration with several network technologies.

The goal of this work was to extend network monitoring to infer about a

possible accident risk even before it is detected by the monitoring system's data analysis center (Monitoring Center) using network applications such as IoT. First experiments using video streaming have shown that the proposed mechanism is feasible and that this technique can represent an improvement in monitoring applications for warning of natural disasters, allowing automatic anticipation of these events. Although some monitoring signals do not express explicit changes in bit rate, for example data from a temperature sensor, an appropriate encoding or modulation can be used to express the value of the data to distinguish when the monitored signal differs from the default value. Future work should extend the simulation by considering BR and CBR streams.

Multiparametric monitoring should increase the accuracy of the predictions. Thus, an EWS based on the approach presented in this paper could use different types of integrated sensors, each transmitted in a unique slice, composing a collection of data to provide better detection models. The threshold of the system is also subject to optimization to obtain a more efficient system.

Future work should explore other ML algorithms, as well as other types of monitoring parameters, that produce different traffic profiles. Although not trivial, the continuity of the ADMITS project in the ITERATION-D project aims to carry out experiments on a real network. To this end, a partnership with Public Safety agencies should be considered to evaluate our solution in risk areas, so that it can be evaluated in depth.

Acknowledgment

This work was partially financed by the projects Stic Amsud - Cooperação em Ciência e Tecnologia da Informação e da Comunicação França - América Do Sul - Capes/Cdefi - 88887.697040/2022-00 (ADMITS) and 88881.985518/2024-01 (ITERATION-D).

References

Afolabi, Ibrahim, et al. "Network slicing & softwarization: A survey on principles, enabling technologies & solutions." IEEE Communications Surveys & Tutorials (2018).

Agarwal, P. K., Efrat, A., Ganjugunte, S. K., Hay, D., Sankararaman, S., & Zussman, G. (2013). The resilience of WDM networks to probabilistic geographical failures. IEEE/ACM Transactions on Networking, 21(5), 1525-1538.

Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., & Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Internet Services and Applications, 9(1), 1-99.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Clayman, S., Neto, A., Verdi, F., Correa, S., Sampaio, S., Sakelariou, I., ... & Serrat, J. (2021). The NECOS approach to end-to-end cloud-network slicing as a service. IEEE Communications Magazine, 59(3), 91-97.

Esposito, M., Palma, L., Belli, A., Sabbatini, L., & Pierleoni, P. (2022). Recent Advances in Internet of Things Solutions for Early Warning Systems: A Review. Sensors, 22(6), 2124.

Esposito, C., Gouglidis, A., Hutchison, D., Gurtov, A., Helvik, B. E., Heegaard, P. E., ... & Rak, J. On the Disaster Resiliency within the Context of 5G Networks: The RECODIS Experience. Institute of Electrical and Electronics Engineers (IEEE), 2018.

Farias, F., Pinheiro, B., Abelém, A., Maciel Jr, P., Rocha, A., Verdi, F., ... & Rothenberg, C. (2019, May). Project NECOS: Towards Lightweight Resource Fatigue in Federated Cloud Infrastructures. In Proceedings of the X Workshop on Experimental Research on the Future Internet (pp. 56-63). SBC.

Foukas, X., Patounas, G., Elmokashfi, A., & Marina, M. K. (2017). Network slicing in 5G: Survey and challenges. IEEE communications magazine, 55(5), 94-100.

Furquim, G., Filho, G. P., Jalali, R., Pessin, G., Pazzi, R. W., & Ueyama, J. (2018). How to improve fault tolerance in disaster predictions: a case study about flash floods using IoT, ML and real data. Sensors, 18(3), 907.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine learning, 63, 3-42.

Gomes, T., Tapolcai, J., Esposito, C., Hutchison, D., Kuipers, F., Rak, J., ... & Tornatore, M. (2016, September). A survey of strategies for communication networks to protect against large-scale natural disasters. In 2016 8th international workshop on resilient networks design and modeling (RNDM) (pp. 11-22) IEEE.

Im, J., Park, H., & Takeuchi, W. (2019). Advances in remote sensing-based disaster monitoring and assessment. Remote Sensing, 11(18), 2181.

Jahir, Y., Atiquzzaman, M., Refai, H., Paranjothi, A., & LoPresti, P. G. (2019). Routing protocols and architecture for disaster area networks: A survey. Ad Hoc Networks, 82, 1-14.

Khan, A., Gupta, S., & Gupta, S. K. (2020). Multi-hazard disaster studies: Monitoring, detection, recovery, and management, based on emerging technologies and optimal techniques. International journal of disaster risk reduction, 47, 101642.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14-76.

Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. Cluster Computing, 22(1), 949-961.

Linardos, V., Drakaki, M., Tzionas, P., & Karnavas, Y. L. (2022). Machine Learning in Disaster Management: Recent Developments in Methods and Applications. Machine Learning and Knowledge Extraction, 4(2), 446-473.

Mijumbi, R., Serrat, J., Gorricho, J. L., Bouten, N., De Turck, F., & Boutaba, R. (2015). Network function virtualization: State-of-the-art and research challenges. IEEE Communications surveys & tutorials, 18(1), 236-262.

Nassif, A. B., Talib, M. A., Nasir, Q., & Dakalbab, F. M. (2021). Machine learning for anomaly detection: A systematic review. Ieee Access, 9, 78658-78700.

Pasquini, R., Miani, R. S., Coelho, P. R., Neto, A. V., Hidalgo, N., Gutiérrez, M., ... & Grampín, E. (2020, June). ADMITS: Architecting Distributed Monitoring and Analytics in IoT-based Disaster Scenarios. In Anais do XII Simpósio Brasileiro de Computação Ubíqua e Pervasiva (pp. 11-20). SBC.

Stadler, R., Pasquini, R., & Fodor, V. (2017). Learning from network device statistics. Journal of Network and Systems Management, 25, 672-698.

Porte, J., Briones, A., Maso, J. M., Pares, C., Zaballos, A., & Pijoan, J. L. (2020). Heterogeneous wireless IoT architecture for natural disaster monitorization.

Rossetto, A., Geyer, C., Arantes, L., and Sens, P.. Impact fd: An unreliable failure detector based on process relevance and confidence in the system. Computer Journal, To be published, 2018.

Scardino, G., Scicchitano, G., Chirivì, M., Costa, P. J., Luparelli, A., & Mastronuzzi, G. (2022). Convolutional Neural Network and Optical Flow for the Assessment of Wave and Tide Parameters from Video Analysis (LEUCOTEA): An Innovative Tool for Coastal Monitoring. Remote Sensing, 14(13), 2994.

Sendai Framework for Disaster Risk Reduction. Available online:

https://www.undrr.org/publication/sendai-frameworkdisaster-risk-reduction-2015-2030 (accessed on 18 November 2024).

Yu, H., Musumeci, F., Zhang, J., Tornatore, M., & Ji, Y. (2020). Isolation-aware 5G RAN slice mapping over WDM metro-aggregation networks. Journal of Lightwave Technology, 38(6), 1125-1137.

Yao, D., Yin, M., Luo, J., & Zhang, S. (2012, November). Network anomaly detection using random forests and entropy of traffic features. In 2012 Fourth International Conference on Multimedia Information Networking and Security (pp. 926-929). IEEE.