Autonomous Network Healing Using AI-driven Diagnostics and Corrective Actions
Term Project | Preliminary Report

By: Gokul Chaluvadi, Abhishek Harish Thumar, Bharath Mahendran

Introduction

The complexity of network infrastructure in distributed environments like edge computing and cloud networks makes manual management both challenging and inefficient. This project, we propose a solution for autonomous network healing that leverages artificial intelligence to diagnose and correct network issues in real-time. This aims to maintain high availability and performance across the distributed networks. This also aligns with the industry trends of shifting towards self-managing networks and it could substantially reduce the operational costs and downtime.

Motivation

Networks often face many problems like intermittent connections, packet loss, high latency, and network failures which cause a lot of downtime. Self-Healing Networks benefit from increased reliability, security, and efficiency. They can reduce the amount of downtime and interruptions faced by standard networks as well. They can predict future issues to help prevent them from happening before they can cause downtime or hardware failures. They are also able to dynamically adjust to the network condition to meet user demand.

Project Assumptions

The project relies on several key assumptions to guarantee its effectiveness and practicality within real-world network environments. First, we assume that the network environments in the system will be handled as distributed and dynamic. This represents the modern cloud and edge computing networks where the diverse nodes have various capabilities. The dynamic nature of networks requires the model to handle rapid changes like conditions and traffic patterns. We have to also assume that historical network data will be available for training our machine learning mode. This allows us to make some baseline patterns of normal network behavior. This data is crucial for the unsupervised model to accurately detect anomalies as deviations from the patterns. Also, our approach assumes that the anomaly detection system must operate in real-time, which signifies low latency detection and correction so we can maintain optimal network performance. Lastly, we have to rely on the NS3 network simulation tool to model realistic network conditions and introduce various types of anomalies. The NS3 provides the necessary granularity to simulate issues like traffic congestion and link failures. This allows us to validate the model's ability to detect and respond to network disruptions in a controlled but realistic environment.

Tools and Technologies

The implementation of this project relies on tools and technologies that have the ability to simulate complex network conditions. Where it can create robust machine learning models and enable real-time network monitoring and diagnosis. First, machine learning models make the core of our anomaly detection and autonomous portion, specifically leveraging unsupervised learning algorithms like autoencoders or clustering for anomaly detection, and reinforcement learning for dynamic, automated corrective actions. To support the development of these models, we will use deep learning libraries like TensorFlow or PyTorch, which offers the flexibility and computational power needed for training and fine-tuning our AI models. For simulation network conditions, we rely on NS3, which is a network simulator tool that allows us to precisely model network scenarios like traffic congestion, link failures, and misconfigurations. NS3's simulation capabilities enable us to recreate complex and realistic network environments, which provide the necessary conditions to test our model under various stress scenarios. Additionally, network monitoring tools will capture the real-time network metrics, which allows us to validate our system's performance against key metrics like detection latency, restoration time, downtime reduction. These tools help to create a framework for building, testing, and refining a self-healing network system that is capable of high availability and performance in distributed network environments.

Autonomous Network Healing Using AI-driven Diagnostics and Corrective Actions
Term Project | Preliminary Report

By: Gokul Chaluvadi, Abhishek Harish Thumar, Bharath Mahendran

Project Framework Overview

The framework for the project, autonomous network healing, focuses on two main tasks, detecting network anomalies and implementing corrective actions in real-time.

<u>Data Collection and Preprocessing:</u> The network traffic data will be collected from a simulated environment. This includes both the normal and abnormal states which are used to train the anomaly detection model.

<u>Anomaly Detection:</u> The unsupervised learning methods, like autoencoders or clustering algorithms, will detect the deviations in the network traffic which indicate anomalies.

<u>Corrective Actions:</u> When we detect an anomaly, the RL model will try to find the best corrective actions based on real-time feedback like rerouting traffic or isolating the malfunctioning nodes.

<u>Simulation and Evaluation:</u> We will use different anomalies simulated in NS3 to evaluate the model's effectiveness in detecting and addressing network issues.

Machine Learning Model

We will use both autoencoders and clustering algorithms for anomaly detection, which leverages their unique strengths for different network conditions. The autoencoders will be used to capture the complex and nuanced patterns in the network traffic by reconstructing the typical traffic behaviors. This allows them to identify the subtle deviations. While clustering groups data into predefined categories based on network activity similarities, which is used as a secondary approach for scenarios where distinct groups of anomalies may arise. This approach ensures accuracy across varying network anomalies.

<u>Unsupervised Learning for Anomaly Detection:</u>
The autoencoders and clustering algorithms are suitable for identifying deviations from normal traffic patterns without requiring the labeled datasets. The models will be trained to recognize typical network conditions, which enables them to spot anomalies in real-time data.

<u>Reinforcement Learning for Corrective Actions:</u>
When the RL detects the anomalies, it will determine the optimal corrective actions based on the real-time feedback. For example, if traffic congestion is detected, the model might reroute traffic or balance loads across the nodes, while learning over time to improve its response.

Datasets

For training the anomaly detection models, we will mainly use historical network data that captures standard operating conditions across various network environments. This data is crucial for training unsupervised learning models like autoencoders and clustering algorithms to recognize normal network behavior patterns.

- <u>MAWI Working Group Traffic Archive</u>: This dataset has extensive network traffic data collected over a series of years, this includes packet traces from real internet traffic in Japan. The dataset provides insights into daily, weekly, and monthly patterns in network usage, which can help the model learn typical fluctuations and identify deviations.

Autonomous Network Healing Using AI-driven Diagnostics and Corrective Actions
Term Project | Preliminary Report
By: Gokul Chaluvadi, Abhishek Harish Thumar, Bharath Mahendran

protocol	packets		bytes		bytes/pkt
total	140507110	(100.00%)	84889763486	(100.00%)	604.17
ip	137407738	(97.79%)	81527070334	(96.04%)	593.32
tcp	105682468	(75.22%)	76787715976	(90.46%)	726.59
http	69496001	(49.46%)	50631764564	(59.64%)	728.56
https	7895980	(5.62%)	5182166004	(6.10%)	656.30
smtp	50847	(0.04%)	11976435	(0.01%)	235.54
ftp	12283	(0.01%)	2393428	(0.00%)	194.86
ssh	780904	(0.56%)	128854815	(0.15%)	165.01
dns	8490	(0.01%)	1645873	(0.00%)	193.86
bgp	5472	(0.00%)	2064069	(0.00%)	377.21
other	27432489	(19.52%)	20826850624	(24.53%)	759.20
udp	8864588	(6.31%)	2623685759	(3.09%)	295.97
dns	480481	(0.34%)	65736525	(0.08%)	136.81
other	8381364	(5.97%)	2554008720	(3.01%)	304.72
icmp	21744790	(15.48%)	1334344327	(1.57%)	61.36
gre	837275	(0.60%)	704213410	(0.83%)	841.08
ipsec	8909	(0.01%)	7875854	(0.01%)	884.03
ip6	269708	(0.19%)	69235008	(0.08%)	256.70
frag	86025	(0.06%)	129732861	(0.15%)	1508.08
ip6	3099370	(2.21%)	3362693032	(3.96%)	1084.96
tcp6	2814817	(2.00%)	3186510623	(3.75%)	1132.05
http	1261596	(0.90%)	1703560163	(2.01%)	1350.32

- CAIDA (Center for Applied Internet Data Analysis): CAIDA provides several network data resources, including anonymized internet traffic data, topology datasets, and network flow data. These datasets are particularly valuable for studying various traffic anomalies and are widely used in network research.

- UNSW-NB15 Dataset: This dataset contains labeled network traffic data collected in a simulated environment. While it includes attack data, which may not be directly relevant to our focus on autonomous healing, its structure and traffic characteristics can still be useful for training anomaly detection models on detecting deviations in network traffic patterns.

CON	0.001055	132	164	31	29	0	0	dns	500473.9375	621800.9375	2	2	0	0	0	0	66	82	0	0	0	0	1421927414	1421927414	0.017	0.013	0	0	0	0	0	0	0	0	3	7	1	3	1	1	1		0
CON	0.036133	528	304	31	29	0	0	-	87676.08594	50480.17188	4	4	0	0	0	0	132	76	0	0	9.89101	10.682733	1421927414	1421927414	7.005	7.564333	0	0	0	0	0	0	0	0	2	4	2	3	1	1	2		0
CON	0.001119	146	178	31	29	0	0	dns	521894.5313	636282.375	2	2	0	0	0	0	73	89	0	0	0	0	1421927414	1421927414	0.017	0.013	0	0	0	0	0	0	0	0	12	8	1	2	2	1	1		0
CON	0.001209	132	164	31	29	0	0	dns	436724.5625	542597.1875	2	2	0	0	0	0	66	82	0	0	0	0	1421927414	1421927414	0.043	0.014	0	0	0	0	0	0	0	0	6	9	1	1	1	1	1		0
CON	0.001169	146	178	31	29	0	0	dns	499572.25	609067.5625	2	2	0	0	0	0	73	89	0	0	0	0	1421927414	1421927414	0.005	0.003	0	0	0	0	0	0	0	0	7	9	1	1	1	1	1		0
CON	0.078339	568	312	31	29	0	0	-	43503.23438	23896.14258	4	4	0	0	0	0	142	78	0	0	29.682221	34.37034	1421927414	1421927414	21.003	24.315	0	0	0	0	0	0	0	0	2	4	2	3	1	1	2		0
CON	0.001134	132	164	31	29	0	0	dns	465608.4688	578483.25	2	2	0	0	0	0	66	82	0	0	0	0	1421927414	1421927414	0.017	0.013	0	0	0	0	0	0	0	0	12	7	1	2	2	1	1		0
INT	0	46	0	0	0	0	0	-	0	0	1	0	0	0	0	0	46	0	0	0	0	0	1421927415	1421927415	0	0	0	0	0	1	2	0	0	0	2	2	2	2	2	2	2		0
CON	0.001126	146	178	31	29	0	0	dns	518650.0938	632326.8125	2	2	0	0	0	0	73	89	0	0	0	0	1421927415	1421927415	0.018	0.013	0	0	0	0	0	0	0	0	6	7	3	1	1	1	1		0
CON	0.001167	132	164	31	29	0	0	dns	452442.1563	562125.0625	2	2	0	0	0	0	66	82	0	0	0	0	1421927415	1421927415	0.018	0.013	0	0	0	0	0	0	0	0	6	7	2	1	1	1	1		0
INT	0	46	0	0	0	0	0	-	0	0	1	0	0	0	0	0	46	0	0	0	0	0	1421927415	1421927415	0	0	0	0	0	1	2	0	0	0	2	2	2	2	2	2	2		0
INT	0	46	0	0	0	0	0	-	0	0	1	0	0	0	0	0	46	0	0	0	0	0	1421927415	1421927415	0	0	0	0	0	1	2	0	0	0	2	2	2	2	2	2	2		0
INT	0	46	0	0	0	0	0	-	0	0	1	0	0	0	0	0	46	0	0	0	0	0	1421927415	1421927415	0	0	0	0	0	1	2	0	0	0	2	2	2	2	2	2	2		0
CON	0.001093	132	164	31	29	0	0	dns	483074.0938	600182.9375	2	2	0	0	0	0	66	82	0	0	0	0	1421927415	1421927415	0.018	0.013	0	0	0	0	0	0	0	0	6	9	2	5	1	1	1		0
CON	0.001851	528	304	31	29	0	0	-	1711507.25	985413.3125	4	4	0	0	0	0	132	76	0	0	0.656903	0.328339	1421927415	1421927415	0.471	0.237667	0	0	0	0	0	0	0	0	8	4	2	5	1	1	2		0
CON	0.001749	528	304	31	29	0	0	-	1811320.75	1042881.625	4	4	0	0	0	0	132	76	0	0	0.640403	0.280968	1421927415	1421927415	0.458333	0.203667	0	0	0	0	0	0	0	0	8	2	3	5	1	1	2		0
CON	0.001128	132	164	31	29	0	0	dns	468085.125	581560.3125	2	2	0	0	0	0	66	82	0	0	0	0	1421927415	1421927415	0.017	0.015	0	0	0	0	0	0	0	0	7	7	2	1	1	1	1		0
CON	0.005153	568	312	31	29	0	0	-	661362.3125	363283.5313	4	4	0	0	0	0	142	78	0	0	1.890104	1.610554	1421927415	1421927415	1.348	1.149333	0	0	0	0	0	0	0	0	8	4	2	5	2	1	2		0
CON	0.004898	568	312	31	29	0	0	-	695794.1875	382196.8125	4	4	0	0	0	0	142	78	0	0	1.780739	1.549507	1421927415	1421927415	1.269667	1.103667	0	0	0	0	0	0	0	0	8	2	3	5	2	1	2		0
CON	0.001111	132	164	31	29	0	0	dns	475247.5313	590459.0625	2	2	0	0	0	0	66	82	0	0	0	0	1421927415	1421927415	0.018	0.013	0	0	0	0	0	0	0	0	5	9	2	1	1	1	1		0
INT	0.000021	728	0	254	0	0	0	-	138666672	0	2	0	0	0	0	0	364	0	0	0	0	0	1421927415	1421927415	0.021	0	0	0	0	0	2	0	0	0	1	1	1	1	1	1	1	Exploits	1
FIN	0.240139	918	25552	62	252	2	10	http	28050.42188	815794.1875	12	24	255	255	1708297952	1939490744	77	1065	1	12026	1170.481668	1144.38336	1421927416	1421927416	21.830818	9.570304	0.051475	0.006528	0.044947	0	1	1	0	0	3	2	2	1	1	1	1	Exploits	1
FIN	2.39039	1362	268	254	252	6	1	http	4233.619141	749.668518	14	6	255	255	3897219059	2466816006	97	45	1	0	18786.7114	941.724938	1421927414	1421927416	183.579303	474.259406	0.066088	0.017959	0.048129	0	1	1	0	0	5	2	2	1	1	1	1	Reconnaissance	1
CON	0.001101	132	164	31	29	0	0	dns	479564.0313	595822	2	2	0	0	0	0	66	82	0	0	0	0	1421927416	1421927416	0.017	0.012	0	0	0	0	0	0	0	0	7	4	3	1	1	1	1		0
CON	0.001082	132	164	31	29	0	0	dns	487985.2188	606284.6875	2	2	0	0	0	0	66	82	0	0	0	0	1421927416	1421927416	0.011	0.009	0	0	0	0	0	0	0	0	7	6	1	1	1	1	1		0
CON	0.001122	132	164	31	29	0	0	dns	470588.2188	584670.1875	2	2	0	0	0	0	66	82	0	0	0	0	1421927416	1421927416	0.018	0.014	0	0	0	0	0	0	0	0	8	3	1	2	2	1	1		0

To evaluate the anomaly detection and corrective action capabilities of our models, we will use the NS3 network simulator. The NS3 allows us to simulate different types of anomalies, like high traffic loads, intentional link disruptions, and misconfigured network paths. By simulating these scenarios, we can create a controlled dataset specifically tailored to the types of challenges our models will encounter in real-world applications. The evaluation dataset will enable us to test the models' performance, accuracy, and response time when dealing with various anomalies in a reproducible setting.

Training and Testing Process

Data Preprocessing:

The first step in preparing our dataset should involve cleaning and normalizing the data collected from the historical datasets. We then have to do preprocessing tasks like removing redundant information, handling missing data, and standardizing feature values (e.g., normalizing traffic volume and latency metrics). This stage is crucial to ensure consistency across datasets and improve the model performance. Additionally, specific feature engineering techniques may be applied to emphasize characteristics like packet delay variations, traffic flow patterns, and routing information. This refined data will be used as the input for training the unsupervised and reinforcement learning models.

Autonomous Network Healing Using AI-driven Diagnostics and Corrective Actions
Term Project | Preliminary Report
By: Gokul Chaluvadi, Abhishek Harish Thumar, Bharath Mahendran

Model Training for Anomaly Detection:
In the next stage, the unsupervised learning model, like an autoencoder or clustering algorithm, is trained to detect deviations from normal network behavior. The model is trained on the preprocessed historical data, which represents normal, anomaly-free traffic patterns. By learning this baseline, the model will be able to identify outliers, flagging them as potential anomalies. Training involves optimizing the model's ability to reconstruct network traffic patterns accurately, allowing it to detect anomalies in real-time once deployed. Techniques like hyperparameter tuning will also be applied to ensure that the model's sensitivity to anomalies aligns with project goals, minimizing false positives while maximizing detection accuracy.

Training the Reinforcement Learning Model for Corrective Actions:
The RL model is trained to automate corrective actions when an anomaly is detected. This process involves setting up a feedback loop where the RL agent interacts with a simulated network environment in NS3. Each time an anomaly is detected, the RL model learns to take appropriate actions, like rerouting traffic, isolating faulty nodes, or balancing traffic loads. Through repeated interaction with the environment, the RL model learns from its successes and mistakes, gradually refining its actions to optimize network performance. The model's reward structure will be based on key metrics like reduced latency, minimized packet loss, and fast restoration times, encouraging the model to prioritize effective and efficient responses.

Simulation Testing and Fine-Tuning in NS3:
Once both models (anomaly detection and RL) are trained, they are tested in a controlled simulation environment using NS3. In NS3, we simulate various network anomalies, like traffic spikes, link disruptions, and misconfigurations, to evaluate the models' responses. These controlled scenarios allow us to assess the system's real-time detection and correction abilities. During testing, we closely monitor key performance indicators like detection latency, restoration time, and network throughput. Any performance issues discovered in this stage will lead to adjustments in model parameters or further training, improving the system's robustness.

Performance Evaluation and Validation:
After simulation testing, we validate the models using an independent dataset of real-world network events or anomalies, if available. This final validation phase is essential for ensuring that the models can generalize beyond the controlled conditions in NS3 and perform accurately in unpredictable network environments. The models are evaluated based on metrics like accuracy, precision, recall (for anomaly detection), and overall network performance improvements (for corrective actions). Through this process, we gain insights into the model's strengths and weaknesses, allowing us to fine-tune the system before deployment.

Next Steps

First, the project will involve selecting and finalizing machine learning models for anomaly detection (like autoencoders or clustering algorithms) and for corrective actions (using reinforcement learning). This will be followed by a data collection and simulation setup phase, where historical network data will be gathered, and the NS3 simulator will be configured to introduce realistic anomaly scenarios. During model development, initial versions of the anomaly detection and corrective action models will be coded and undergo preliminary testing to ensure baseline functionality. Once the models are functioning, iterative testing within the NS3 environment will allow for the fine-tuning of model parameters and enhancements to accuracy, detection latency, and response times, ultimately aiming to maximize system performance and reliability.

Autonomous Network Healing Using AI-driven Diagnostics and Corrective Actions
Term Project | Preliminary Report
By: Gokul Chaluvadi, Abhishek Harish Thumar, Bharath Mahendran

References and Related Work

Fang, H., Yu, P., Tan, C., Zhang, J., Lin, D., Zhang, L., Zhang, Y., & Li, W. (2024). Self-Healing in Knowledge-Driven Autonomous Networks. IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/10562327

Ghosh, S., & Ghosh, S. (2009). A survey on self-healing systems: Approaches and systems. ACM Computing Surveys, 42(2), 1-37. https://doi.org/10.1145/1592451.1592453

Ayodi, S., Adesanya, O., & Wu, L. (2023). A survey of machine learning techniques for detecting anomaly in internet of things (IoT). International Journal of Machine Learning and Cybernetics. https://doi.org/10.1007/s13042-023-01664-y

Zhao, Y., Gong, W., & He, X. (2020). A survey of anomaly detection techniques for IoT in industries. IEEE. Retrieved from https://ieeexplore.ieee.org/document/9089465

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. Retrieved from https://arxiv.org/pdf/1707.06347

Qin, Y., & Zhang, S. (2022). Survey of anomaly detection in IoT: Applications and challenges. arXiv preprint. https://arxiv.org/pdf/2211.11949