УДК: 004.896
DOI 10.33514/1694-7851-2025-3/1-589-596

Лю Фан
Ph.D докторант
И. Арабаев атындагы Кыргыз мамлекеттик университети
Бишкек ш.
Ли Дань
Ph.D докторант
И. Арабаев атындагы Кыргыз мамлекеттик университети
Бишкек ш.
Чжэн Сяохан
Ph.D докторант
И. Арабаев атындагы Кыргыз мамлекеттик университети
Бишкек ш.
328146066@qq.com

PYTHON ЖАНА ЖАСАЛМА ИНТЕЛЛЕКТТИН НЕГИЗИНДЕ ТАРМАКТЫК БАШКАРУУ СИСТЕМАСЫН ДОЛБООРЛОО ЖАНА ИШКЕ АШЫРУУ

Аннотация. Азыркы тармактык башкаруу системаларынын автоматташтыруу деңгээли, каталарга жооп берүү ылдамдыгы жана кол менен техникалык тейлөө чыгымдары сыяктуу проблемаларын эске алуу менен, бул макалада Python жана жасалма интеллект технологияларына негизделген акылдуу тармактык башкаруу системасын сунуштадык жана ишке ашырдык. Бул система браузер/сервер (B/S) архитектурасын колдонот, Python тилинин ийкемдүүлүгүн жана жасалма интеллект алгоритмдеринин алдыңкылыгын бириктирип, тармактык жабдууларды автоматтык түрдө башкаруу, акылдуу катаны диагностикалоо жана эскертүү, маалыматтарды терең анализдөө сыяктуу функцияларды ишке ашырат. Эксперименттик натыйжалар көрсөткөндөй, бул система тармакты башкаруунун эффективтүүлүгүн айтарлыктай жогорулатат, техникалык тейлөө чыгымдарын төмөндөтөт жана тармак администраторлорго эффективтүү жана акылдуу башкаруу куралын камсыз кылат.

Негизги сөздөр: Тармактык башкаруу системасы, автоматташтыруу деңгээли, каталарга жооп берүү ылдамдыгы, кол менен техникалык тейлөө чыгымдары, Python, жасалма интеллект технологиялары, B/S архитектурасы, акылдуу катаны диагностикалоо, эскертүү.

Лю Фан
Ph.D докторант
Кыргызский государственный университет имени И. Арабаева
г. Бишкек
Ли Дань
Ph.D докторант
Кыргызский государственный университет имени И. Арабаева
г. Бишкек
Чжэн Сяохан
Ph.D докторант
Кыргызский государственный университет имени И. Арабаева

г. Бишкек

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ СЕТЬЮ НА ОСНОВЕ PYTHON И ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Аннотация. В связи с текущими проблемами в области автоматизации, эффективности реагирования на сбои и высокой стоимости ручного обслуживания существующих систем управления сетями, в данной статье предлагается и реализуется интеллектуальная система управления сетью на основе Python и технологий искусственного интеллекта. Система использует архитектуру браузер/сервер (B/S), сочетая гибкость языка Python и передовые алгоритмы искусственного интеллекта, для реализации таких функций, как автоматизированное управление сетевым оборудованием, интеллектуальная диагностика и предупреждение неисправностей, а также углублённый анализ данных. Результаты экспериментов показывают, что данная система значительно повышает эффективность управления сетью, снижает эксплуатационные расходы и предоставляет сетевым администраторам эффективный и интеллектуальный инструмент управления.

Ключевые слова: система управления сетью, уровень автоматизации, эффективность реагирования на сбои, стоимость ручного обслуживания, Python, технологии искусственного интеллекта, B/S-архитектура, интеллектуальная диагностика неисправностей, предупреждение.

Liu Fang
Ph.D student
Kyrgyz state university named after I. Arabaev
Bishkek c.
Li Dan
Ph.D student
Kyrgyz state university named after I. Arabaev
Bishkek c.
Zheng Xiaohang
Ph.D student
Kyrgyz state university named after I. Arabaev
Bishkek c.

DESIGN AND IMPLEMENTATION OF A NETWORK MANAGEMENT SYSTEM BASED ON PYTHON AND ARTIFICIAL INTELLIGENCE

Abstract. Addressing the current bottlenecks in network management systems in terms of automation level, fault response efficiency, and manual operation and maintenance costs, this paper proposes and implements an intelligent network management system based on Python and artificial intelligence technology. The system adopts a browser/server (B/S) architecture, combining the flexibility of the Python language with the advancement of artificial intelligence algorithms to achieve functions such as automated management of network devices, intelligent fault diagnosis and early warning, and in-depth data analysis. Experimental results show that the system can significantly enhance network management efficiency, reduce operation and maintenance costs, and provide network administrators with efficient and intelligent management tools.

Keywords: Network management system, Automation level, Fault response efficiency, Manual operation and maintenance cost, Python, Artificial intelligence technology, B/S architecture, Intelligent fault diagnosis, Early warning.

Introduction

With the rapid development of information technology, the network has become a crucial infrastructure for the operation of modern society. However, the expansion of network scale and the increase in system complexity have made traditional manual network management models difficult to cope with dynamic changes and complex issues. Therefore, promoting the development of network management systems towards intelligence has become an important trend in the current technological field.

Python, with its concise syntax, robust data processing capabilities, and extensive database support, is widely used in artificial intelligence and network management. Artificial intelligence technologies, such as machine learning and deep learning, have demonstrated immense potential in network traffic analysis, fault prediction, and status monitoring, significantly enhancing the level of intelligence in network management. This study integrates Python with artificial intelligence technologies to design and implement an intelligent network management system.

The system adopts a Browser/Server (B/S) architecture and is developed based on the Python Django framework, enabling functions such as automated management, real-time monitoring, timely alerts, intelligent data analysis, and automatic fault handling for network devices. The system aims to provide network administrators with efficient and intelligent management tools, enhancing network management efficiency and reliability, and reducing operation and maintenance costs.

1 System architecture design

This intelligent network management system adopts a Browser/Server (B/S) architecture, with the overall structure divided into three major parts: front end, back end, and database, supplemented by multiple functional function libraries to support system operation. The front end is responsible for presenting and interacting with the user interface, developed using technologies such as HTML5, CSS3, and JavaScript, ensuring an intuitive and user-friendly interface with convenient operation. The back end, as the core of the system, is developed using Python and is responsible for processing business logic and algorithm implementation. The database part uses MySQL as the main database, and utilizes SQLAlchemy as an ORM tool to achieve efficient data persistence operations without directly writing SQL. The front-end interface development is based on the Vue.js framework, leveraging its two-way data binding and state management mechanism to optimize the user interaction experience and enhance development efficiency and code reusability.

The frontend communicates with the backend through a RESTful API and utilizes the Axios library to handle HTTP requests. The backend is developed using the Django framework, leveraging its MTV (Model-Template-View) architecture pattern and Django REST Framework to facilitate rapid construction and maintenance of API interfaces. The database design employs MySQL to store structured data such as network device information, user data, and log records, while introducing Redis as a caching database to optimize the read and write performance of frequently accessed data. Data transmission adopts a lightweight JSON format and ensures transmission security through the HTTPS protocol. The backend integrates the Celery framework for asynchronously executing time-consuming tasks such as device status monitoring and fault alerts, effectively enhancing system response speed.

In terms of function libraries, the system primarily relies on the following Python libraries: ① Numpy and Pandas for data cleaning and analysis; ② Scikit-learn for implementing machine learning models; ③ TensorFlow for training and inference of deep learning models; ④ Paramiko for SSH connections and network device management; ⑤ PSutil for system resource monitoring; ⑥ Matplotlib and Seaborn for data visualization and display. The integration of these libraries not only enhances development efficiency but also ensures the overall performance and stability of the system.

2. Division of functional modules in network management system

As a key tool for enhancing the efficiency of network management and maintenance, the functional modules of a network management system must be designed to comprehensively cover the daily maintenance needs of network management. The main functional modules of a network management system are divided as shown in Figure 1.

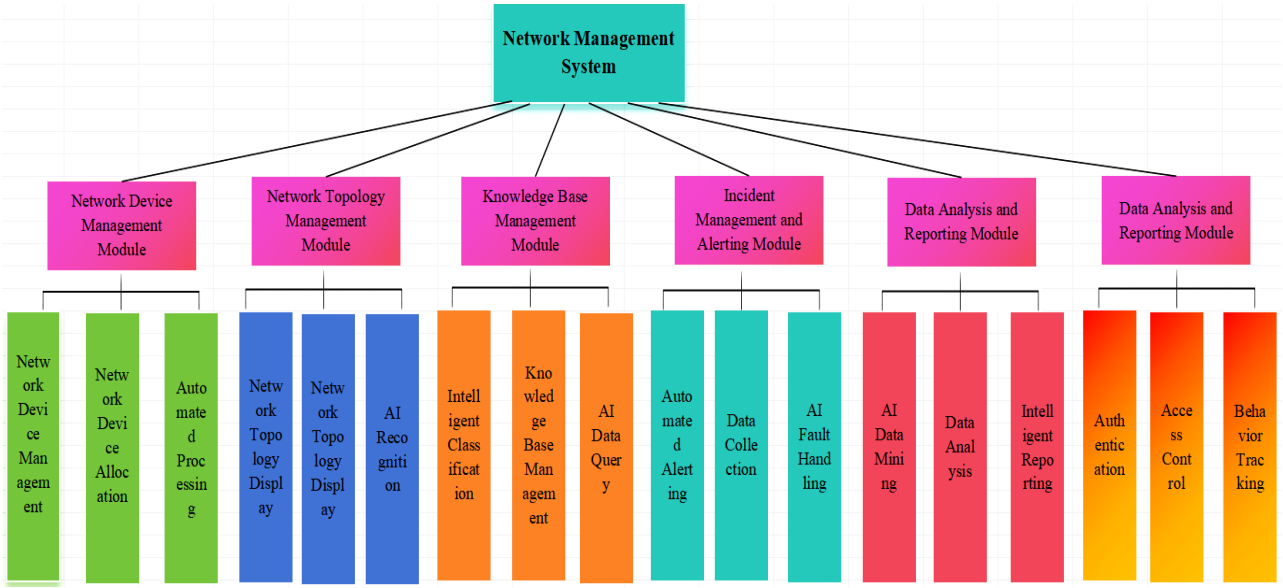

Figure 1: Design of functional modules of network management system

2.1 Network device management module

The network device management module is the core component of this system, primarily responsible for operations such as adding, deleting, modifying, and querying network devices. By integrating SNMP and SSH protocols, this module achieves real-time monitoring and management of network devices. Adopting an object-oriented design philosophy, the module abstracts network devices as classes, encapsulating their basic attributes and operation methods. Through a combination of regular polling and event-driven methods, it obtains real-time device status information such as CPU utilization, memory usage, and interface traffic. Additionally, this module provides device configuration backup and recovery functions to ensure the security and recoverability of network device configurations.

2.2 Network topology management module

The network topology management module is responsible for automatically discovering and visually displaying the network topology structure. This module constructs a network topology diagram by automatically identifying network devices and their connection relationships, utilizing the LLDP and SNMP protocols to achieve automatic discovery and visual display of the network topology structure. D3.js is a JavaScript-based data-driven document manipulation library that provides a rich set of components and tools for creating highly customized interactive data visualizations. The module supports dynamic display and interaction of the topology diagram,

including functions such as zooming, dragging, and viewing device information, and allows administrators to manually adjust the topology structure. In addition, the module can detect anomalies in the network, such as loops and islands, and promptly issue alerts to help administrators quickly locate network issues.

2.3 Knowledge base management module

The knowledge base management module is used to store and manage knowledge and experience related to network management. This module employs Neo4j graph database to store knowledge data. Through reasonable table structure design, index optimization, and query adjustment, it utilizes its powerful relationship processing capabilities to ensure efficient data storage and retrieval. The module supports the storage of various knowledge types such as fault handling schemes, device configuration templates, and network optimization suggestions. It also understands user query intentions through natural language processing technology and provides accurate knowledge recommendations. Additionally, this module possesses self-learning capabilities, which enable it to extract effective knowledge from historical fault handling records and continuously enrich the content of the knowledge base.

2.4 Fault Handling and Early Warning Module

The fault handling early warning module serves as the intelligent core of this system, primarily responsible for predicting, detecting, and handling network faults. This module employs machine learning algorithms, utilizing random forests to compare and rank the importance of fused features, effectively eliminating redundant features and constructing new fused features, thus preparing for fault recognition by the LSTM neural network. By analyzing the status data and historical fault data of network equipment, a fault prediction model is established. When potential faults are detected, the module automatically triggers an early warning and searches for a handling solution in the knowledge base. For common faults, the module can automatically execute predefined repair scripts to achieve automated fault handling. Additionally, the module provides a fault tracking function, recording the entire process of fault handling for subsequent analysis and optimization.

2.5 Data analysis and reporting module

The data analysis and reporting module undertakes the tasks of collecting, parsing, and visually presenting network operation data. This module acquires information from multiple data sources such as network devices and traffic monitoring systems, and utilizes Pandas and Numpy to complete data cleaning and preprocessing. Through visualization tools such as Matplotlib and Seaborn, it generates various charts such as traffic trend charts and device status charts. The module also supports custom report generation, enabling administrators to flexibly construct various statistical reports according to actual needs. Furthermore, this module embeds machine learning algorithms, which can monitor abnormal behaviors in network traffic in real time, identify potential security risks and performance bottlenecks, and provide data support for network optimization.

2.6 User permission and authentication module

The user permission and authentication module bears the core responsibility of system security management and access control. Based on the Role-Based Access Control (RBAC) model, this module provides fine-grained configuration of operational permissions for various users. By introducing the JSON Web Token (JWT) mechanism, it authenticates and authorizes user identities, ensuring the security of system access. The module incorporates built-in user management functions, supporting operations such as user creation, deletion, and permission adjustments. To further enhance system security, the module also integrates a two-factor authentication mechanism.

Additionally, the system comprehensively records all user operational behaviors and generates audit logs, providing a reliable basis for subsequent security reviews and traceability.

3 Implementation methods of key technologies

Intelligent automatic processing technology based on equipment status and fault recurrence is one of the core supporting technologies of this system. This technology constructs equipment status trend prediction models and fault pattern intelligent recognition models through in-depth analysis of historical operating status and fault logs of network equipment, thereby achieving early warning and automated handling of potential faults.

Firstly, the system collects various operational status parameters of network devices, including CPU usage, memory occupation, interface traffic, etc., and uses them as input variables for the device status prediction model. A Long Short-Term Memory (LSTM) neural network is employed to model these time series data, in order to infer the future operational trends of the devices. Once the predicted values exceed the preset alarm threshold, the system will automatically trigger the early warning mechanism, prompting the administrator to intervene in a timely manner, thereby effectively avoiding potential risks.

In terms of fault handling, the system adopts a hybrid approach combining rule-driven and machine learning techniques. By analyzing historical fault data, typical fault features are extracted and processing rules are constructed, which are stored in a knowledge base. When the system detects an abnormal state of the equipment, it performs fault pattern matching in the knowledge base based on the current state features and automatically invokes the corresponding processing script for response. For new types of faults that occur for the first time, the system records and analyzes them using machine learning algorithms to continuously optimize the processing strategy and enhance the system's adaptability.

This article also introduces a reinforcement learning mechanism, combining deep neural networks and convolutional neural networks to extract features and recognize patterns from collected data. The system can continuously learn and self-optimize during fault handling, receiving reward or punishment feedback based on the handling effectiveness (such as time consumption, success rate, etc.), thereby dynamically adjusting its decision-making strategy. This method effectively enhances the system's adaptability to changes in the network environment, significantly improving the efficiency and accuracy of fault handling.

The fault handling and early warning module serves as the core safeguard mechanism of the intelligent system. AI technology has constructed a "perceptive brain" for the communication network, enabling it to swiftly identify abnormal signals, precisely pinpoint the source of faults, and facilitate automatic problem resolution based on vast historical experience. This self-intelligent network architecture has boosted fault localization efficiency by 80%, shortened decision-making and handling time by 75%, and significantly reduced the duration of user perception of faults. Simultaneously, the system employs automated testing and monitoring tools to periodically inspect critical metrics such as network connectivity status and server response time. Upon detecting issues, it can automatically initiate the repair process or send alert messages, thereby significantly enhancing the stability and reliability of the network.

4 System verification

4.1 Setup of system experimental environment

4.1.1 Hardware Environment

Server: Dell PowerEdge R740, equipped with 2 Intel Xeon processors (10 cores, 2.2GHz). Memory: 64GB. Storage: 1TB SSD (system disk) + 4TB HDD (data storage). Network equipment:

3 Huawei series switches, 2 Huawei AR1200 routers. Terminal equipment: 10 ThinkStation workstations (for simulating user terminals).

4.1.2 Software Environment

Operating System: Ubuntu Server 20.04 LTS. Virtualization Platform: Docker 20.10.12 + Kubernetes 1.22. Databases: MySQL 8.0.27 (primary database), Redis 6.2.6 (cache database), Neo4j 4.4.3 (knowledge graph database). Development Environment: Python 3.9.7, Django 4.0.3, Vue.js 3.2.31. AI Frameworks: TensorFlow 2.7.0, Scikit-learn 1.0.2.

4.1.3 Network topology environment

The experimental network adopts a three-layer architecture. Core layer: two routers form a redundant architecture. Aggregation layer: three switches form a ring topology. Access layer: 10 VLANs are divided, with 10 to 15 terminal devices connected to each VLAN.

4.2 Selection of test data

Test data is primarily obtained through the following three channels: (1) Utilizing Python scripts to simulate and generate device operation status data, accounting for 40% of the total data volume; (2) Artificially injecting five types of typical network fault data, including link interruptions, device overloads, configuration errors, etc., accounting for 30%; (3) Collecting actual fault data in the constructed network topology environment, accounting for 40%.

4.3 Experimental methods

The experimental setup includes three comparative groups: ① the traditional method comparison group, utilizing a monitoring system based on SNMP polling and a fault handling system based on a rule engine; ② the improved method comparison group, employing a prediction model solely based on LSTM and a classification model solely based on random forest, respectively; ③ the intelligent network management system, which constructs a hybrid model of LSTM and random forest and introduces reinforcement learning optimization strategies. The comparative results of the three experimental groups are presented in Table 1.

Table 1 Comparison of Fault Detection Effectiveness of Three Methods

Metric	conventional methods	Improved methods	The present system
Accuracy / %	71.5	85.2	92.9
Recall / %	68.8	82	88.9
F1-score	0.705	0.835	0.912
MTTD/s	27.9	12.1	5.6

The analysis results indicate that: ① this system significantly outperforms other comparative methods in various performance indicators; ② the hybrid model effectively combines the advantages of temporal features and classification features; ③ the reinforcement learning mechanism enables the system to quickly adapt to new fault patterns.

The advantage analysis is as follows: ① The multi-model fusion strategy significantly improves the accuracy of fault identification; ② The combination of knowledge base and automatic processing mechanism greatly reduces the fault recovery time; ③ The microservice architecture provides strong support for the scalability of the system.

4.4 Typical fault handling cases

For example, in the case of an abnormal surge in traffic on a certain switch port, the system detected that the traffic exceeded the preset threshold by 30%, and immediately automatically searched the knowledge base to identify three potential causes of the failure. By running a diagnostic script, the

system ultimately confirmed that the anomaly was caused by a broadcast storm, and immediately automatically shut down the relevant port while triggering the alarm mechanism. The entire processing flow took only 8.3 seconds, which is a significant improvement in efficiency compared to the traditional manual processing that requires more than 15 minutes.

5 Conclusion

The main innovations of this system are reflected in the following aspects: ① Leveraging the efficient development capabilities of the Python language and its rich ecosystem resources, a fully functional network management system has been successfully constructed. ② Deep integration with artificial intelligence technology has enabled multiple intelligent functions, including fault prediction and automatic handling. ③ The adoption of a modular architecture design has significantly improved the scalability and maintainability of the system. ④ Through a knowledge base and self-learning mechanism, the system is equipped with the ability to continuously optimize and evolve itself. Experimental results show that the system excels in enhancing network management efficiency and reliability, as well as reducing operation and maintenance costs. It particularly demonstrates significant advantages in fault prediction and automated processing.

Of course, the network management system itself is highly complex, and there is still much room for optimization and improvement. In the future, we can further optimize the algorithm model to enhance prediction accuracy; expand system compatibility to support more types of network devices; and strengthen system security to cope with increasingly complex cybersecurity threats.

Overall, this article proposes a practical and feasible technical solution for intelligent network management systems, laying an important foundation for the subsequent development of this field. With the continuous evolution of artificial intelligence technology and the increasing complexity of the network environment, network management systems based on Python and artificial intelligence will play an increasingly crucial role in future network operation and maintenance.

REFERENCES

1. Yu Xiaoyong, Qin Liyun, Gui Haitao, Ou Shifeng, Wu Lifang. Application of New-generation Artificial Intelligence in Intelligent Perception and Fault Diagnosis of Distribution Network [J]. Southern Power Grid Technology, 2022, 16 (05): – 34-43.
2. Wei Zhijun. Method for secure transmission and sharing of network information data in a big data environment based on cloud computing [J]. Information and Computer (Theoretical Edition), 2023, 35 (11): – 233-235.
3. Guo Wei. Research on Python Teaching Strategies in the Context of Artificial Intelligence. Computer Knowledge and Technology, 2023, 16 (31): – 174-176.
4. Zhang Yuewen. Design and Implementation of a Wired Communication Network Management System Based on SNMP [D]. Nanchang: Nanchang University, 2023.
5. Cao Ying. Research and Practice on the Design of Digital Twin Smart Industrial Visualization Display [D]. Nanjing: Nanjing University of Technology, 2024.
6. Zhang Ding. Method for secure data transmission in wireless communication networks based on adversarial encryption [J]. Yangtze Information and Communication, 2023, 36 (12): – 181-183.
7. Zhang Yuewen. Design and Implementation of a Wired Communication Network Management System Based on SNMP [D]. Nanchang: Nanchang University, 2023.
8. Tan Xin. Data Security Transmission Method for the Operation of Power Grid Trading Websites [J]. Computer Programming Techniques and Maintenance, 2025(1): – 90-92, 133.

Рецензент: кандидат технологических наук Бийбосунова С.К.