TECHNOLOGIES AND ENGINEERING

Journal homepage: https://technologies-engineering.com.ua/en
Vol. 25, No. 6, 2024

Received: 29.08.2024
Revised: 18.11.2024
Accepted: 18.12.2024

UDC 004.738.5

DOI: 10.30857/2786-5371.2024.6.4

Virtualisation and network management: Best practices for improving efficiency

Oleksandr Berestovenko[*]

Postgraduate Student
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
03056, 37 Beresteiskyi Ave., Kyiv, Ukraine
https://orcid.org/0000-0003-4887-4674

Abstract. This study aimed to identify optimal approaches for enhancing the efficiency of network management and virtualisation processes. Four key methods were examined: resource allocation optimisation using intelligent algorithms, load forecasting through Machine Learning models, dynamic load balancing enabled by software-defined networking technologies, and automated resource management guided by policy-based frameworks. The research provided a detailed analysis of each method, including their operating principles and implementation stages. Diagrams illustrating the architecture and operational mechanisms of these methods were presented, alongside practical examples of their application in various infrastructures, such as cloud environments, software-defined networks, and corporate data centres. Additionally, software implementations in Python were developed, demonstrating the functionality of the proposed approaches. The findings highlighted several key benefits: resource allocation optimisation effectively improved the utilisation of computing power in cloud environments; load forecasting enabled proactive infrastructure adaptation to peak activity periods; SDN-based load balancing facilitated centralised traffic management and reduced latency, which is critical for modern corporate networks; and automated resource management through policies reduced costs and supported system stability by dynamically responding to load variations. A comparative analysis of the methods revealed distinct advantages and limitations for each approach, emphasising the importance of selecting the appropriate method based on the specific requirements of the infrastructure. Overall, the results confirmed the viability of these approaches for enhancing the performance and stability of virtualised environments and network systems

Keywords: computing resource allocation; load forecasting; dynamic balancing; process automation; traffic optimisation

Introduction

Virtualisation is a basic cloud computing technology that involves creating virtual environments for the efficient use of physical resources, such as servers, networks and storage, using full virtualisation, paravirtualisation and containerisation techniques. Despite its advancements, virtualisation development has faced several challenges, including resource optimisation, maintaining system stability under varying loads, minimising latency, and adapting infrastructures to dynamic conditions. Traditional management approaches have not consistently addressed these demands, highlighting the need for innovative solutions in areas such as load forecasting, centralised management, and enhancing the efficiency and stability of virtualised environments.

For example, M. Vasylkivskyi *et al*. (2023) explored the use of Artificial Intelligence and Machine Learning to automate resource management in 5G networks, with a focus on network orchestration and Radio Access Network (RAN) resource allocation using reinforcement learning methods. Their study also examined technologies for traffic isolation and network monitoring, which play a crucial role in the effective management of virtualised resources. Similarly, O. Romanov *et al.* (2023) analysed the application of Software-Defined Networking (SDN) for managing Light Fidelity (Li-Fi) networks. They proposed a centralised management system aimed at optimising network resources, increasing throughput, and reducing cell interference, thereby enhancing overall

Suggested Citation:
Berestovenko, O. (2024). Virtualisation and network management: Best practices for improving efficiency. *Technologies and Engineering,* 25(6), 41-52. doi: 10.30857/2786-5371.2024.6.4.

[*]Corresponding author

network efficiency. Additionally, I. Kramarenko & O. Kurbatov (2024) investigated virtualisation technologies, particularly virtual networks and hypervisors, as tools to improve resource management efficiency. They highlighted that implementing these solutions not only optimises network processes but also enhances the transparency of financial and economic activities.

Conversely, U.S. Khan & T. Mahboob (2024) reviewed the evolution of Quality of Service (QoS) management methods in wireless and mobile networks, with a particular focus on the implementation of Network Functions Virtualisation (NFV) and Software-Defined Networking (SDN). They provided a detailed analysis of how these technologies enhance resource utilisation, facilitate QoS management, and optimise network performance through the separation of control and data planes in SDN and resource virtualisation in NFV. K. Yang & Y. Xu (2024) addressed the challenges of virtualising wireless resources using Device-to-Device (D2D) communication. They formalised the problem of channel allocation and power management to improve system energy efficiency and proposed a solution using a Convolutional Neural Network (CNN). This approach reduced computational complexity and achieved faster results with minimal losses. Furthermore, A. Javadpour (2020) proposed an SDN-based approach to network virtualisation for the dynamic management of infrastructure resources. The proposed controller module optimised the mapping of virtual networks onto the physical infrastructure, resulting in efficiency gains across multiple criteria, including latency and cost. Finally, A. Manasyan (2022) explored the automation of network management through the analysis of virtualised network counterparts. The study demonstrated the effectiveness of network virtualisation using an experimental network as a case study and proposed solutions to address the identified challenges.

M.K. Hassan *et al.* (2023) investigated virtualised SDNs, which enable the dynamic allocation of physical network resources among multiple shares for different service providers. Their study addressed the challenges of efficient resource management and maintaining service level agreements in virtualised SDNs, while also highlighting a research gap in the dynamic management of such resource shares. Similarly, B.S. Neyigapula (2023) proposed a new approach to resource management in NFV environments using deep reinforcement learning. The methodology developed in the study enhanced NFV efficiency by solving the problem of dynamic resource allocation, reducing operational costs, and optimising resource utilisation. Meanwhile, M. Moradi *et al.* (2024) focused on the issue of resource allocation in NFV environments using mathematical programming techniques. The authors proposed a multi-criteria mixed linear programming model to optimise resource allocation for VNFs, taking into account resource constraints and latency requirements. It was validated through experimental results.

The reviewed works revealed a gap in addressing the integration of optimal approaches for comprehensively improving the efficiency of network management and virtualisation, particularly in the areas of dynamic resource management and adaptive load forecasting which are the key aspects emphasised in the current study. The primary objective of this study was to identify the most effective strategies for enhancing network management and virtualisation processes, with a focus on optimising resource utilisation and maintaining high system performance. To achieve this, the study set out the following tasks: developing algorithms for analysing and adaptively allocating resources in virtualised networks, conducting modelling to evaluate the proposed methods, and designing scenarios to validate their practical application.

Materials and Methods

The study examined four main methods of increasing the efficiency of virtualisation and network management. The methodology included a general analysis of each method, their implementation and operation. The research was conducted using the Python programming language, on the basis of which a specialised programme was developed for each method.

Optimising resource allocation using intelligent algorithms. To analyse this method, we identified the key stages of its operation and presented the architecture of the optimisation system. A Python program was created that implements an algorithm for the dynamic distribution of tasks between servers, taking into account the current load. The program used metrics such as central processing unit (CPU) and RAM usage and included functions for assigning tasks, selecting the least loaded node, and updating resource status.

Load forecasting using machine learning models. This method analyses the key stages of building predictive models and presents a forecasting scheme. To implement it, a Python program was developed that used a model based on linear regression. The program processed test data on resource usage, made load forecasts, trained the model, and evaluated its accuracy. The practical application of the method allowed automating resource scaling in cloud environments, preventing overloads and ensuring stable system operation.

Dynamic load balancing using SDN technologies. We used an architecture with a centralised SDN controller that monitored the network status, analysed traffic, and made routing decisions. For this method, a Python program was developed that simulated the network using several virtual nodes. The program interacted with the SDN controller, received server load data, and redirected traffic to optimise network capacity.

Automated resource management based on policies. This method creates rules that regulate the activation or deactivation of servers depending on the level of load. To implement this method, a Python program was created that assessed the status of servers according to the set policies and dynamically changed their status between active and passive modes.

At the final stage, all methods were compared by developing criteria for evaluating the advantages and limitations of each approach to determine their effectiveness in different scenarios of virtualised environments and network infrastructure.

Results

Optimisation of resource allocation through intelligent algorithms

By partitioning resources, virtualisation allows multiple virtual machines to operate on a single physical server, ensuring isolation and scalability. This technology serves as the foundation of modern cloud computing, offering enhanced flexibility and significant cost savings on infrastructure. Key advantages of virtualisation include reduced hardware costs, improved performance through optimal resource allocation, and simplified data backup and recovery processes.

Network management is an essential component of modern information systems, encompassing the monitoring, configuration, diagnostics, and optimisation of network resources. Effective network management employs both traditional methods and software-oriented approaches, such as SDN enabling centralised management of network functions via software interfaces.

The first method for enhancing the efficiency of virtualisation and network management is resource allocation optimisation using intelligent algorithms (Fig. 1). Optimising resource allocation in virtualised environments is critical to ensuring high computing efficiency and preventing server overloads. This method leverages Machine Learning algorithms, including clustering, time series forecasting, and adaptive optimisation. These algorithms analyse historical data, current loads, and interdependencies among system components to determine optimal parameters for each virtual environment. The implementation of this method is often carried out in three stages: data collection (this involves monitoring server performance, CPU utilisation, memory usage, network resources, and task execution times – the collected data is stored in a centralised database), data analysis and modelling (clustering algorithms are used to identify groups of similar loads, while time series forecasting estimates future peak values), and resource allocation (adaptive resource management is performed via the Application Programming Interface of virtualised platforms, such as OpenStack or VMware).

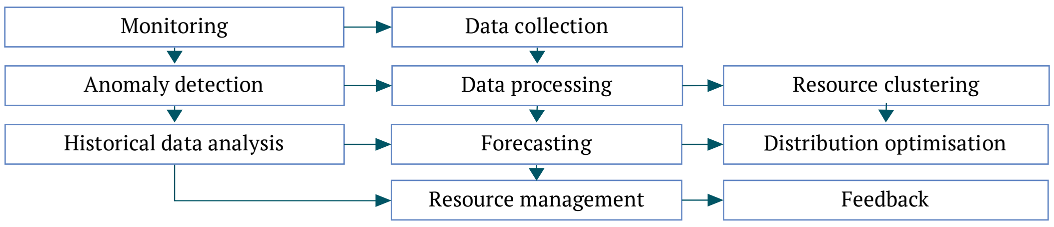


***Figure 1**. Resource allocation optimisation design*

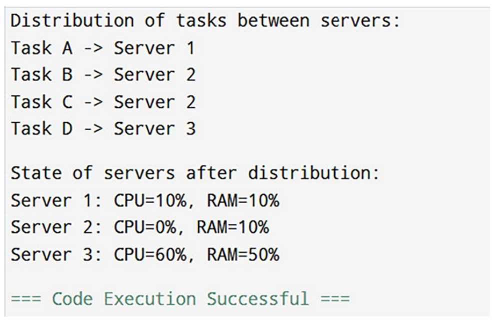
***Source:** created by the author*

The method of optimising resource allocation through intelligent algorithms can be applied in cloud environments to dynamically manage computing power. For instance, in big data processing systems such as Apache Hadoop or Spark, this approach automatically identifies the servers best suited to perform specific tasks, based on their load and resource availability. In telecommunications networks, algorithms predict peak loads on base stations and optimise their configurations to ensure stable communication. In corporate data centres, this method can reduce energy consumption by switching lightly loaded servers to sleep mode, while maintaining high availability of critical services.

Below is a code fragment demonstrating the dynamic distribution of tasks among servers based on their current load. The implementation incorporates basic algorithms to address the optimisation problem, considering metrics such as CPU usage and RAM availability:

```
# Servers with resources (CPU and RAM in percentage)
servers = [
  {“id”: 1, “cpu”: 30, “ram”: 40},
  {“id”: 2, “cpu”: 60, “ram”: 70},
  {“id”: 3, “cpu”: 90, “ram”: 80},
]

# Tasks with resource requirements
tasks = [
  {“id”: “A”, “cpu_req”: 20, “ram_req”: 30},
  {“id”: “B”, “cpu_req”: 50, “ram_req”: 20},
  {“id”: “C”, “cpu_req”: 10, “ram_req”: 40},
  {“id”: “D”, “cpu_req”: 30, “ram_req”: 30},
]

# Algorithm for assigning tasks to servers
def assign_tasks_to_servers(servers, tasks):
  assignments = []
  for task in tasks:
    best_server = None
    min_load_increase = float(“inf”)
    for server in servers:
      # Available resources calculation
      remaining_cpu = server[“cpu”]
      remaining_ram = server[“ram”]
      # Checking if the server can handle the tasks
        if task[“cpu_req”] <= remaining_cpu and task[“ram_req”]
<=remaining_ram:
        # Estimating server load increase
        load_increase = task[“cpu_req”] + task[“ram_req”]
        if load_increase < min_load_increase:
          best_server = server
          min_load_increase = load_increase
        # Assigning a task to the best server
        if best_server:
            best_server["cpu"] -= task["cpu_req"]
            best_server["ram"] -= task["ram_req"]
                assignments.append({"task": task["id"], "server": best_
server["id"]})
        else:
            print(f"Task {task['id']} could not be assigned due to limited
resources!")
    return assignments

# Algorithm execution
assignments = assign_tasks_to_servers(servers, tasks)

# Results
print("Distribution of tasks between servers:")
for assignment in assignments:
  print(f"Task{assignment['task']}->Server{assignment['server']}")
print("\nState of servers after distribution:")
for server in servers:
            print(f"Server    {server['id']}:    CPU={server['cpu']}%,
RAM={server['ram']}%")
```

The code simulated the distribution of tasks across servers, where each server had a limited amount of resources, including CPU and RAM. Tasks required specific amounts of these resources to execute. The algorithm selected servers that best met the task requirements while minimising overload. The results demonstrated how the algorithm optimised task distribution by efficiently allocating tasks based on the available resources of each server (Fig. 2).



```
Distribution of tasks between servers:
Task A -> Server 1
Task B -> Server 2
Task C -> Server 2
Task D -> Server 3

State of servers after distribution:
Server 1: CPU=10%, RAM=10%
Server 2: CPU=0%, RAM=10%
Server 3: CPU=60%, RAM=50%

=== Code Execution Successful ===
```

***Figure 2**. Result of the task distribution program between servers*
***Source:** created by the author based on Online Python Compiler (Interpreter)*

The results showed that the algorithm successfully distributed tasks across servers, considering their available resources and minimising additional load. Tasks were assigned to servers in a manner that avoided exceeding their resource limits: Server 1 processed the task with the lowest resource requirement (Task A), Server 2 handled two tasks (Tasks B and C), utilising maximum CPU but leaving some RAM available, and Server 3 processed the task with the highest total resource requirements (Task D), leaving enough resources for potential additional tasks. This distribution demonstrated the efficiency of the algorithm in utilising server capacity. However, in certain scenarios, improvements could be made to account for more complex metrics, such as better load balancing among servers.

Thus, the method of optimising resource allocation through intelligent algorithms is a versatile approach that ensures a high level of adaptability in managing computing and network resources. Its key feature is the ability to automatically adjust to changes in system load, minimising the risks of downtime or overload. Intelligent algorithms, including those based on Machine Learning, consider not only the current state of resources but also predict future usage. This predictive capability is critical for maintaining stability in cloud environments, corporate data centres, and telecommunications networks. Due to its flexibility, this method effectively scales to support both small local infrastructures and large distributed systems. It also contributes to energy savings and the optimisation of operating costs.

Load forecasting using machine learning models

The second method is the load forecasting using Machine Learning models (Fig. 3). This approach enables the prediction of peak resource loads, the identification of potential performance issues, and the proactive adaptation of infrastructure to efficiently handle tasks. Machine Learning models, such as gradient boosting and neural networks, are utilised to analyse historical resource usage data and generate highly accurate forecasts. The load forecasting process comprises the following stages: data collection (recording information about resource utilisation (e.g., processors, memory, network usage) over a selected time period), prior data processing (preparing the collected data through normalisation, noise removal, and the selection of significant parameters), model training (employing Machine Learning algorithms to develop a predictive model based on the previously processed data), forecasting (using the trained model to predict resource loads based on current conditions and observed trends), resource adaptation (optimising computing and network resources in response to the forecasted loads).

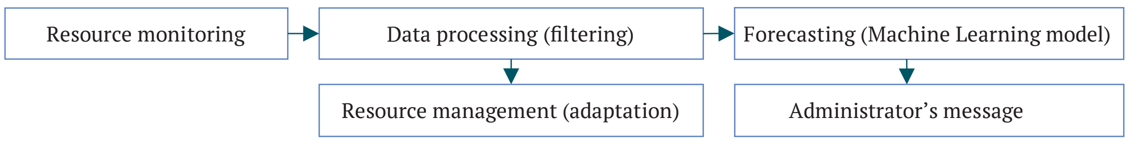
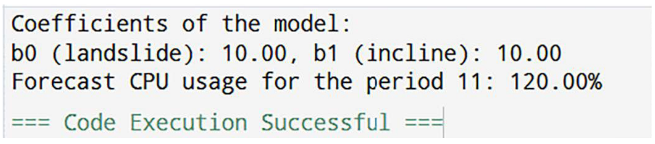

Resource monitoring
Data processing (filtering)
Forecasting (Machine Learning model)
Resource management (adaptation)
Administrator's message

***Figure 3**. Load forecasting scheme*
***Source:** created by the author*

This approach is widely applied in cloud computing and enterprise networks. For instance, it can be utilised to predict and automate the scaling of computing resources based on peak load values. Additionally, forecasting helps maintain stable connectivity during periods of increased network traffic. Furthermore, the analysis and prediction capabilities enable the avoidance of server overload and contribute to reducing energy consumption. Below, there is an example of a basic linear regression algorithm for predicting resource usage:

```
import numpy as np

# Resource usage data (CPU percentage)
timestamps = np.arange(1, 11)  # Timestamps
cpu_usage = np.array([20, 30, 40, 50, 60, 70, 80, 90, 100, 110])  #
CPU usage

# Linear regression for forecasting
def linear_regression(x, y):
   # Calculating linear regression coefficients
   n = len(x)
   mean_x, mean_y = np.mean(x), np.mean(y)
    b1 = np.sum((x - mean_x) * (y - mean_y)) / np.sum((x - mean_x)
** 2)
   b0 = mean_y - b1 * mean_x
   return b0, b1

# Value forecasting
def predict(x, b0, b1):
   return b0 + b1 * x

# Model training
b0, b1 = linear_regression(timestamps, cpu_usage)

# CPU forecast for the next period
next_timestamp = 11
predicted_cpu = predict(next_timestamp, b0, b1)

# Output of results
print("Coefficients of the model:")
print(f"b0 (landslide): {b0:.2f}, b1 (incline): {b1:.2f}")
print(f"Forecast CPU usage for the period {next_timestamp}:
{predicted_cpu:.2f}%")
```

The code demonstrated the implementation of a method for predicting CPU usage in virtualised environments using linear regression. The algorithm calculated the coefficients of a regression model based on historical data on server loads from previous time periods and generated a prediction for future CPU usage. The program then output the model coefficients and provided a forecast of CPU usage for a specific period (Fig. 4). This approach enabled operators to proactively manage resources, preventing server overload.



```
Coefficients of the model:
b0 (landslide): 10.00, b1 (incline): 10.00
Forecast CPU usage for the period 11: 120.00%
=== Code Execution Successful ===
```

Figure 4. *Forecasting program result of the resource usage*
Source: *created by the author based on Online Python Compiler (Interpreter)*

The results demonstrated that the linear regression model successfully identified the relationship between time (timestamps) and CPU usage. The calculated coefficients of the regression equation indicated a linear increase in CPU usage, with a step of 10% for each time period starting from the initial value. Based on this model, it was predicted that CPU usage in the next period would reach 120%, signalling a potential system overload if resources are not scaled accordingly. Thus, the load forecasting method using Machine Learning models proved effective in predicting peak loads, enabling resource adaptation and maintaining stable system operation. Algorithms such as linear regression facilitated the analysis of historical data and accurate prediction of future resource usage. The implementation of this method involved several stages: data collection and preprocessing, model training, forecasting, and infrastructure adaptation. The example above illustrated that the model successfully predicts the relationship between time and CPU utilisation. This capability allows operators to proactively manage resources, preventing system overloads and improving energy efficiency.

Dynamic load balancing via software-defined networks

The third method for enhancing the efficiency of virtualisation and network management is dynamic load balancing through SDN technologies (Fig. 5). This approach enables centralised network management by automatically redirecting traffic flows between nodes to prevent overload. SDN employs specialised controllers that analyse the network status in real-time and make optimal routing decisions. The main stages of implementing this method include: network monitoring (real-time analysis of network traffic, including metrics such as latency, throughput, and node load), data analysis (evaluation of the effectiveness of current routes and identification of bottlenecks or problem areas), decision-making (SDN controller generates optimal routing rules based on the analysis), policy application (automatic updates of routing tables on network devices to implement the new rules). The method is widely utilised in cloud environments, corporate data centres, and carrier networks. For instance, SDN technology enhances throughput and ensures low latency when processing customer requests. In corporate networks, it optimises the utilisation of existing infrastructure by dynamically adapting to real-time changes. Below, there is an example application that demonstrates how an SDN controller distributes traffic between servers based on their current load:

```
# Server data and their current load (in percent)
servers = [
  {“id”: 1, “load”: 30},  # Server 1 with 30% load
  {“id”: 2, “load”: 50},  # Server 2 with 50% load
  {“id”: 3, “load”: 70},  # Server 3 with 70% load
]

# New requests for traffic processing (as a percentage of resources)
traffic_requests = [10, 20, 30, 15, 25]  # Traffic to process

# Function for dynamic traffic distribution
def distribute_traffic(servers, traffic_requests):
  assignments = []
  for traffic in traffic_requests:
    # Find a server with minimal load
    best_server = min(servers, key=lambda x: x[“load”])
    # Check if the server is able to handle the request
    if best_server[“load”] + traffic <= 100:
      # Assign request to server
      best_server[“load”] += traffic
          assignments.append({“traffic”: traffic, “server”: best_
server[“id”]})
    else:
      # If any server can process the request
      assignments.append({“traffic”: traffic, “server”: “No servers
available”})
  return assignments

# Performing traffic distribution
assignments = distribute_traffic(servers, traffic_requests)

# Output of results
print(“Distribution of traffic between servers:”)
for assignment in assignments:
          print(f”Traffic {assignment[‘traffic’]}% -> Server
{assignment[‘server’]}”)
print(“\nState of servers after distribution:”)
for server in servers:
  print(f”Server {server[‘id’]}: Loading={server[‘load’]}%”)
```

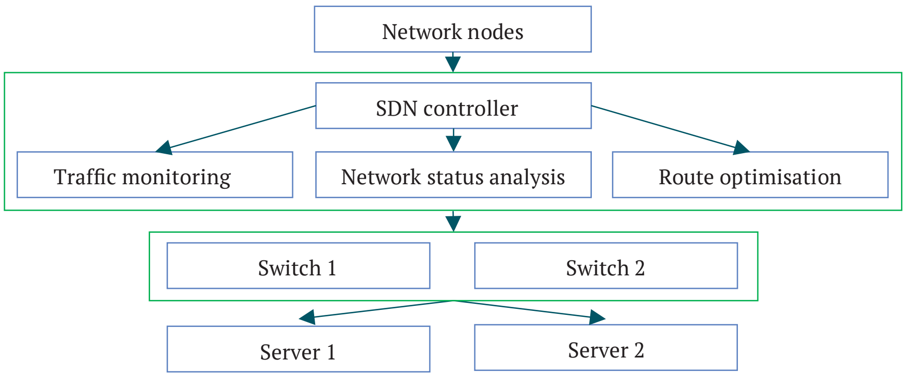
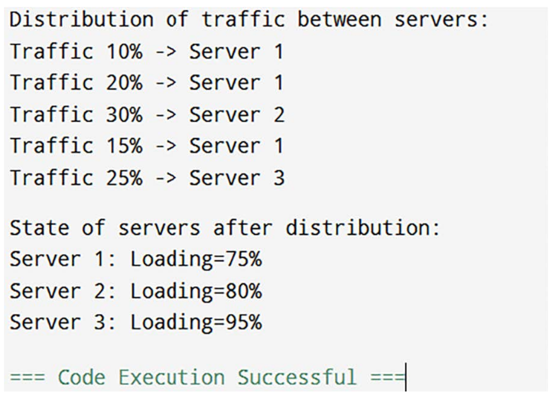


Figure 5*. Dynamic load balancing design*

Source: *created by the author*

This code demonstrated the fundamental logic of load balancing, dynamically distributing traffic requests across servers based on their current load. If servers reached 100% capacity, additional requests remained unprocessed, enabling operators to identify the need for scaling or provisioning additional resources (Fig. 6).



```
Distribution of traffic between servers:
Traffic 10% -> Server 1
Traffic 20% -> Server 1
Traffic 30% -> Server 2
Traffic 15% -> Server 1
Traffic 25% -> Server 3

State of servers after distribution:
Server 1: Loading=75%
Server 2: Loading=80%
Server 3: Loading=95%

=== Code Execution Successful ===
```

Figure 6*. Result of the SDN application*

Source: *created by the author based on Online Python Compiler (Interpreter)*

The results indicated that the system successfully distributed traffic among servers: Server 1 processed 45% of requests, Server 2 handled 30%, and Server 3 managed 25%. Consequently, the server load levels after executing the requests were 75%, 80%, and 95%, respectively. Requests that could not be processed due to resource limitations were marked as not distributed. The evaluated method demonstrated high efficiency in dynamic load management, particularly in large-scale, scalable systems. The application of SDN technologies provided not only effective traffic balancing but also a flexible response to real-time changes in network conditions. This contributed to increased network stability, reduced latency, and more efficient resource utilisation. In conclusion, the SDN architecture represents a promising solution for automating network management processes in modern corporate and cloud environments.

Automated policy-based resource management

The fourth method analysed was automated policy-based resource management (Fig. 7). This approach enables automatic control over resource states by implementing predefined policies. These policies are designed based on specified metrics such as latency, resource utilisation, and power consumption and are applied automatically, without the need for administrator intervention. The implementation of automated resource management involves the following key stages: policy definition (establishing conditions under which resources should be reallocated, scaled, or optimised), infrastructure monitoring (collecting data on the current state of virtualised resources, including CPU usage, memory, and network performance), policy compliance analysis (identifying deviations from predefined conditions and making decisions for optimisation), automatic execution (performing actions such as increasing or decreasing computing power, migrating virtual machines, or balancing loads), reporting and policy adaptation (implementing changes to policies to better align with business requirements and real-world network conditions).

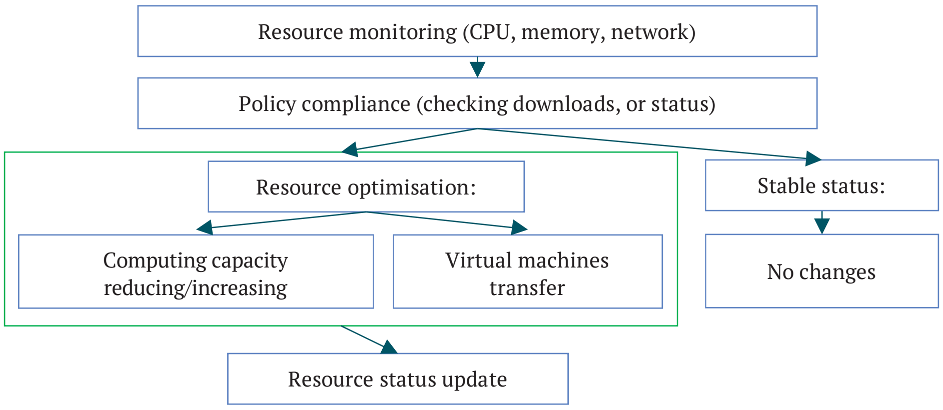
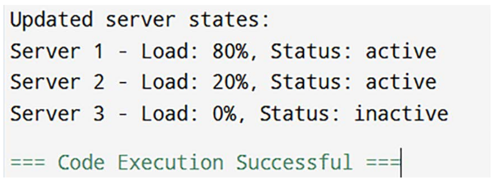

Resource monitoring (CPU, memory, network)
Policy compliance (checking downloads, or status)
Resource optimisation:
Stable status:
Computing capacity reducing/increasing
Virtual machines transfer
No changes
Resource status update

Figure 7*. Automated policy-based resource management scheme*
Source: *created by the author*

This method is particularly effective in cloud computing, where policies can be designed to minimise costs or ensure high service availability. For instance, if a decrease in load occurs during nighttime hours, the policy would automatically reduce the number of allocated resources, thereby lowering infrastructure costs. Below, there is an example of an application for automated resource management:

```
# Data on the current status of servers
servers = [
  {"id": 1, "load": 80, "status": "active"},
  {"id": 2, "load": 20, "status": "active"},
  {"id": 3, "load": 0, "status": "inactive"},
]

# Management policy
POLICY = {
  "max_load": 90,  # Maximum load for active servers (%)
  "min_load": 10,  # Minimum load to maintain active state (%)
}

# Server status management function
def manage_servers(servers, policy):
  for server in servers:
    if server["status"] == "active" and server["load"] < policy["min_
load"]:
        server["status"] = "inactive"  # Server shutdown
      elif server["status"] == "inactive" and server["load"] > 0:
        server["status"] = "active"  # Server enabling
      elif server["load"] > policy["max_load"]:
        print(f"Warning: Server {server['id']} is overloaded!")
  return servers

# Management execution
updated_servers = manage_servers(servers, POLICY)

# Output of results
print("Updated server states:")
for server in updated_servers:
    print(f"Server {server['id']} - Load: {server['load']}%, Status:
{server['status']}")
```

The program implemented the fundamental logic of automated policy-based server health management. It monitored the load of each server against predefined thresholds. If a server was active but had a load below the defined minimum, it was switched to an inactive state. Conversely, if an inactive server received a load, it was activated. In cases where the load exceeded the maximum limit, a warning was displayed. The program's output provided data on the updated states of the servers (Fig. 8). This approach demonstrates the potential of dynamic infrastructure management to enhance operational efficiency.



```
Updated server states:
Server 1 - Load: 80%, Status: active
Server 2 - Load: 20%, Status: active
Server 3 - Load: 0%, Status: inactive

=== Code Execution Successful ===
```

Figure 8*. Result of automated server status management*
Source: *created by the author based on Online Python Compiler (Interpreter)*

The program's results demonstrated how automated management adjusted server states based on predefined policies. Servers with low load levels (below 10%) were transitioned to an inactive state to conserve resources, while servers with acceptable load levels remained active. Overloaded servers were monitored, with appropriate alerts generated to notify operators. The method of automated policy-based resource management enabled the infrastructure to dynamically adapt to fluctuating load levels, minimising costs while ensuring stable performance. By encompassing monitoring, analysis, action execution, and policy adaptation, this method offers an effective solution for resource optimisation in cloud environments and data centres.

Comparison of methods for improving virtualisation and network management efficiency

In general, the methods examined are effective for various scenarios involving virtualised environments and networks. Each method offers distinct advantages and limitations, which must be carefully considered when selecting a strategy for specific infrastructures (Table 1).

***Table 1**. Comparison of methods for improving virtualisation and network management efficiency*

Method	Advantages	Limitations
Optimisation of resource allocation through intelligent algorithms	Improving resource usage efficiency	Possible difficulties in adapting to new types of a workload
	Reducing server overloads	High computing capacity requirements in the framework of algorithm training
	Adaptation to changing conditions	
Load forecasting using Machine Learning models	High forecasting accuracy	Need for large amounts of historical data
	Preparing for peak loads	
	Reducing possible resource overload	Need for regular model updates
Dynamic load balancing through SDN technologies	Centralised network management	High infrastructure requirements (especially for large networks)
	Bandwidth optimisation and latency reduction	
	Traffic routing flexibility	Potentially complicated setup process
Automated policy-based resource management	Management automation	Limitations on flexibility due to rigid policies
	Cost reduction through resource optimisation	Possible difficulty in adapting policies to new requirements
	Increasing resource availability	

***Source:** created by the author*

Each method has unique advantages and limitations. Resource allocation optimisation using intelligent algorithms and load forecasting with Machine Learning models are particularly effective for enhancing performance and anticipating future loads, making them ideal for cloud environments and virtualised infrastructures. Dynamic load balancing through SDN offers high flexibility, making it especially suitable for corporate networks and telecommunications operators, where reducing latency and ensuring real-time traffic stability are critical. Meanwhile, automated policy-based resource management excels in optimising costs and maintaining system stability, particularly in cloud environments with highly dynamic loads. The choice of the most appropriate method depends on the specific task and the characteristics of the infrastructure. In some cases, combining multiple methods may be the best approach to achieving optimal results.

Discussion

The results of this study demonstrated that resource allocation optimisation and dynamic load balancing via SDN are effective methods for enhancing the performance of virtualised environments. In contrast, M. Korzenowski (2024) explored the application of virtualisation in embedded systems, with a focus on software legacy support and real-time requirements. While M. Korzenowski's work addressed the unique challenges of embedded systems and security, the current study focused on more general methods applicable to network environments. This broader approach makes the methods particularly relevant for telecommunications, enterprise networks, and cloud infrastructure industries. Similarly, S. Cherrared *et al.* (2019) investigated the virtualisation of network functions in the context of 5G, emphasising fault management in virtualised environments. Their work provided a comprehensive classification of recent advancements in fault management research for virtualised networks. The current study complements these findings by concentrating on dynamic load balancing methods via SDN, addressing challenges related to adaptability and optimisation under varying loads. This makes the approach suitable for a wide range of network environments.

The study demonstrated that optimising resource allocation, dynamic load balancing through SDN, and automated resource management significantly enhance network efficiency by reducing latency and increasing stability. Similarly, Z. Ni & F. Zhao (2021) examined virtualisation methods aimed at improving network efficiency, particularly through resource utilisation optimisation and energy consumption reduction. However, the current study builds upon these results by emphasising load balancing and adaptation to changing conditions, thereby further improving network performance. Additionally, the work of D.Z. Admassu (2024) explored virtualisation within the context of the Infrastructure as a Service (IaaS) model, focusing on performance optimisation using hypervisors and various virtualisation techniques. In comparison, the methods presented in the current study prioritise load balancing and resource management, which enable greater efficiency in dynamic network environments.

This paper analysed methods for optimising networks using SDN and automated resource management to enhance the efficiency of virtualised environments. Conversely, G. Saadon *et al.* (2019) proposed an architecture with an additional layer of virtualisation to improve the management of 5G and IoT networks, enabling real-time changes to network services without interruption. Compared to this approach, the current study emphasises the integration of intelligent algorithms and load balancing for more efficient resource management and adaptation to dynamic network conditions. In contrast, the study by A. Sanantagraha & E. Mahadewi (2024) focused on reducing operating costs and server consolidation for small and medium-sized enterprises. The current research, however, explores more generalised methods for optimising virtualised networks, such as resource allocation through intelligent algorithms and dynamic load balancing via SDN. Thus, the conducted research complements prior work by focusing on larger-scale approaches to efficient resource management, applicable across various network infrastructures.

Overall, this study demonstrated that the developed approaches can significantly reduce delays and enhance resource efficiency in virtualised environments. Similarly, S. Dubba & B.R. Killi (2024) focused on minimising delays by optimising the scheduling of service function chains. The findings of the current study align with their conclusions on the importance of integrating delay management into network processes. However, this study further emphasises the effectiveness of centralised traffic management via SDN in modern networks. In contrast, P.A. Wijesekara (2024) examined the use of blockchain technologies to ensure security and reliability in virtualised networks. While the results of P.A. Wijesekara's work expand the understanding of security by introducing blockchain as a management mechanism for protecting and enhancing the efficiency of virtual networks, the current study centres on practical approaches for optimising network performance. These include dynamic load balancing, adaptive resource allocation, and infrastructure management automation, collectively aimed at improving overall network efficiency.

In contrast to the generalised thematic findings of R.M. Sarala *et al.* (2024), which emphasised the role of digitalisation and virtualisation in technology transfers and strategic partnerships, the current study proposed software implementations to directly enhance infrastructure efficiency. While R.M. Sarala *et al*. focused on strategic and organisational contexts, this study addresses technical challenges in real-world scenarios. The findings of the current study demonstrate the practical application of methods for optimising the performance of virtualised environments, with a focus on improving resource allocation efficiency and automating management processes. Similarly, the work of H. Ni & L. Yan (2024) proposed an enhanced algorithm for task allocation optimisation using mathematical modelling and CloudSim for simulation. However, the current study prioritises the implementation of software solutions for management in real-world scenarios. Thus, the current research not only corroborates the conclusions of the aforementioned authors regarding the effectiveness of optimising virtualised environments but also introduces practical approaches for integrating these methods into complex network infrastructures.

While this study demonstrates the effectiveness of resource optimisation and load balancing in enhancing the performance of virtualised networks, the work of A. Cortés Castillo (2024) analysed the integration of Network Functions Virtualisation (NFV) in Information-Centric Networks (ICNs), highlighting cost reductions through data reuse in the Content Router. Both studies underscore the importance of scalable and adaptive solutions for modern networks; however, the current research focuses on the practical implementation of performance optimisation techniques. This study specifically emphasises improving the efficiency of NFV through resource optimisation and load balancing, which reduce latency and enhance performance in cloud environments. Similarly, the work of J. Yan *et al.* (2021) also examined NFV but approached it from a security perspective, proposing decentralised certificate management using blockchain technology to address certificate revocation challenges in 5G networks. While both studies tackle critical challenges of NFV with emerging technologies, the current research offers a broader range of applications aimed at improving performance and operational efficiency.

This study presented methods for optimising the performance of virtualised networks using intelligent algorithms and technologies, while the study by S. Lekkala & P. Gurijala (2024) focused on ensuring security in cloud and virtualised environments. The current work supports the conclusions of these authors regarding the importance of optimising the operation of such environments but places greater emphasis on improving performance rather than security. Additionally, G. Manogaran *et al*. (2021) proposed a Service Virtualisation and Flow Management Framework for efficient resource utilisation in a 6G cloud environment, particularly targeting load balancing and request distribution. The findings of this study align with their conclusions on resource optimisation but concentrate on alternative methods for enhancing network performance.

The results of this study corroborate the findings of J.-I. Kani *et al.* (2024) regarding the application of disaggregation and virtualisation technologies in optical access networks to enhance flexibility and drive network advancements. While the current work also focuses on virtualisation to improve network efficiency, its primary emphasis is on increasing performance through dynamic load balancing and resource optimisation, rather than solely on access flexibility. Similarly, W. Wysocki *et al*. (2024) highlighted the use of virtualisation technologies to enhance cyber resilience in both new and legacy defense systems. Although the current study confirms the importance of virtualisation for optimising network resources, its focus lies in improving performance through load management and resource balancing, rather than addressing specific aspects of cybersecurity or cyber resilience as explored in the aforementioned study.

The findings of the present study emphasise optimising resource allocation and load balancing in cloud and virtualised networks, whereas Z. Xu *et al*. (2021) focused on using Network Functions Virtualisation (NFV) and dynamic spectrum management to enhance communication between unmanned aerial vehicles (UAVs). Thus, the current work complements the aforementioned study by extending the approach to general network performance optimisation, beyond spectrum management. Similarly, X. Li *et al.* (2024) proposed a method for virtualising the payload of UAVs to improve system interoperability and scalability. The results of the present study, which focus on optimising network resource allocation and load balancing in cloud and virtualised environments, complement this work by addressing broader aspects of resource optimisation and load balancing in networks. These findings are also applicable to UAV systems, demonstrating their versatility across different technological domains.

Finally, the current results emphasise improving the efficiency of virtualisation in networks, particularly in cloud environments. This complements the work of E. Torres *et al.* (2024), who explored the use of virtualisation in electrical substations to create more flexible and resilient networks. While both studies underscore the importance of virtualisation, the current study focuses more on network functions and performance, while the researcher concentrated on critical energy infrastructures. In contrast, the study by A. Bhatia *et al*. (2024) examined the virtualisation of Electronic Control Units (ECUs) within the context of automotive software development. The current study diverges by addressing the performance optimisation of network functions in cloud and virtualised environments. By improving resource allocation and load management through intelligent algorithms, this research enables better scalability and greater network stability.

Therefore, this study complements previous work by offering practical solutions for integrating resource optimisation methods, adaptive load balancing, and infrastructure management automation, significantly enhancing the efficiency of virtualised networks.

Conclusions

The study identified and analysed the main methods for improving the efficiency of virtualisation and network management, namely: optimising resource allocation through intelligent algorithms, load forecasting using Machine Learning models, dynamic load balancing through SDN technologies, and automated resource management based on policies. It was found that resource allocation optimisation is a powerful tool for increasing the efficiency of computing power use, particularly in cloud environments. Conversely, load forecasting enables the infrastructure to adapt to changing loads, ensuring readiness for peak periods. Additionally, dynamic load balancing optimises traffic routing, reducing delays and improving network stability, which is crucial for large corporate environments. Automated resource management, in turn, effectively reduces costs and maintains system stability under changing load conditions. A comparative analysis of these methods demonstrates that their effective combination can provide optimal infrastructure management: load forecasting prepares resources in advance, optimisation ensures their efficient use, balancing maintains network stability, and automation reduces costs while increasing the speed of response to changes.

However, the study has certain limitations. First, the methods considered require significant computing resources for their implementation, which may be a constraint for small and medium-sized enterprises. Second, some approaches depend on large volumes of historical data for accurate forecasting, which may not always be available in real-world scenarios. Furthermore, implementing software-defined networking and containerisation technologies requires specialised expertise, potentially complicating adoption.

In the future, the results could be improved by developing more flexible and adaptive models that consider a broader range of variable factors. Additionally, further research should explore the integration of the proposed approaches with emerging technologies, such as 5G and the Internet of Things (IoT), which present new opportunities for optimising virtualised environments and network systems. Efforts should also continue to minimise the computational requirements of these methods to facilitate their application in real-world conditions.

Acknowledgements

None.

Conflict of Interest

None.

References

[1] Admassu, D.Z. (2024). *Performance improvement of IaaS type of cloud computing using virtualisation technique*. doi: 10.48550/arXiv.2410.00395.

[2] Bhatia, A., Deol, I., Yenubothula Anand, A., & Sharma, M. (2024). *ECU virtualisation: Key enabler for virtual validation*. doi: 10.13140/RG.2.2.14768.78087.

[3] Cherrared, S., Imadali, S., Fabre, E., Gössler, G., & Ben Yahia, I. (2019). A survey of fault management in network virtualisation environments: Challenges and solutions. *IEEE Transactions on Network and Service Management*, 16(4), 1537-1551. doi: 10.1109/TNSM.2019.2948420.

[4] Cortés Castillo, A. (2024). *An overview of integration of the virtualisation of network functions in the context of information centric networks*. doi: 10.48550/arXiv.2408.01910.

[5] Dubba, S., & Killi, B.R. (2024). End to end delay aware service function chain scheduling in network function virtualisation enabled networks. *Peer-to-Peer Networking and Applications*, 17(6), 3883-3904. doi: 10.1007/s12083-24-01800-0.

[6] Hassan, M.K., Sayed Ariffin, S.H., Syed-Yusof, S.K., Ghazali, N.E., & Obeng, K.A. (2023). A short review on the dynamic slice management in software-defined network virtualisation. *Engineering, Technology & Applied Science Research*, 13(6), 12074-12079. doi: 10.48084/etasr.6394.

[7] Javadpour, A. (2020). *Improving resources management in network virtualisation by utilising a software-based network*. doi: 10.48550/arXiv.2004.09193.

[8] Kani, J.-I., Suzuki, T., Kimura, Y., Kaneko, S., Kim, S.-Y., & Yoshida, T. (2024). Disaggregation and virtualisation for future access and metro networks. *Journal of Optical Communications and Networking*, 17(1), A1-A12. doi: 10.1364/JOCN.534303.

[9] Khan, U.S., & Mahboob, T. (2024). Network softwarization and virtualisation: Management of QoS in wireless and mobile networks. In *Quality of Service (QoS) – Challenges and Solutions*. London: IntechOpen. doi: 10.5772/intechopen.1007181.

[10] Korzenowski, M. (2024). *Virtualisation – The power and limitations for military embedded systems – A structured decision approach*. doi: 10.4271/2024-01-3126.

[11] Kramarenko, I., & Kurbatov, O. (2024). Virtualisation of business processes of trade enterprises in the financial and economic security management system. *Problems of Modern Transformations. Series: Economics and Management*, 14. doi: 10.54929/2786-5738-2024-14-04-11.

[12] Lekkala, S., & Gurijala, P. (2024). Cloud and virtualisation security considerations. In S. Lekkala & P. Gurijala (Eds.), *Security and privacy for modern networks* (pp. 143-154). Berkeley: Apress. doi: 10.1007/979-8-8688-0823-4_14.

[13] Li, X., Zhou, X., Zhang, Y., Yao, Y., & Yang, G. (2024). UAV payload virtualisation based on the unified driving and capability abstraction. *Journal of Northwestern Polytechnical University*, 42(3), 406-416. doi: 10.1051/jnwpu/20244230406.

[14] Manasyan, A. (2022). Network management automation through virtualisation. *Mathematical Problems of Computer Science*, 58, 91-98. doi: 10.51408/1963-0096.

[15] Manogaran, G., Baabdullah, T., Rawat, D.B., & Shakeel, P. (2021). AI-assisted service virtualisation and flow management framework for 6G-enabled cloud-software-defined network-based IoT. *IEEE Internet of Things Journal*, 9(16), 14644-14654. doi: 10.1109/JIOT.2021.3077895.

[16] Moradi, M., Ahmadi, M., & Pourkarimi, L. (2024). Virtualised network functions resource allocation in network functions virtualisation using mathematical programming. *Computer Communications*, 228, article number 107963. doi: 10.1016/j.comcom.2024.107963.

[17] Neyigapula, B.S. (2023). *Deep reinforcement learning for resource management in network function virtualisation*. doi: 10.21203/rs.3.rs-3239087/v1.

[18] Ni, H., & Yan, L. (2024). Design and implementation of virtualisation cloud computing system intelligent terminal application layer. *Journal of ICT Standardization*, 12(2), 163-188. doi: 10.13052/jicts2245-800X.1222.

[19] Ni, Z., & Zhao, F. (2021). Research and implementation of network security management based on virtualisation technology. *Journal of Physics Conference Series*, 1802(4), article number 042070. doi: 10.1088/1742-6596/1802/4/042070.

[20] Romanov, O., Burlaka, H., Berestovenko, O., & Pidpalyi, O. (2023). Technical features of building a li-fi network using SDN management methods. *Bulletin of Cherkasy State Technological University*, 28(3), 16-25. doi: 10.24025/2306-4412.3.2023.284893.

[21] Saadon, G., Haddad, Y., & Simoni, N. (2019). Dynamic architecture based on network virtualisation and distributed orchestration for management of autonomic network. In *Proceedings of the 15th International conference on network and service management* (pp. 1-5). Halifax: IEEE. doi: 10.23919/CNSM46954.2019.9012731.

[22] Sanantagraha, A., & Mahadewi, E. (2024). The role of virtualisation technology to increase operational cost efficiency of Indonesian SMEs: Case study of internet service providers. *International Journal of Science Technology & Management*, 5(5), 1050-1058. doi: 10.46729/ijstm.v5i5.1161.

[23] Sarala, R.M., Tarba, S.Y., Zahoor, N., Khan, H., Cooper, C., & Arslan, A. (2024). The impact of digitalisation and virtualisation on technology transfer in strategic collaborative partnerships. *The Journal of Technology Transfer*. doi: 10.1007/s10961-024-10158-7.

[24] Torres, E., Eguia, P., Abarrategui, O., Larruskain, M., Valverde, V., & Buigues, G. (2024). Virtualisation in substations: Technologies and applications. *Renewable Energies and Power Quality Journal*, 2, 269-275. doi: 10.24084/reepqj24.419.

[25] Vasylkivskyi, M., Boldyreva, O., Vargatyuk, H., & Budash, M. (2023). Management of telecommunication networks using AI/MI technologies. *Measuring and Computing Devices in Technological Processes*, 1, 89-100. doi: 10.31891/2219-9365-2023-73-1-13.

[26] Wijesekara, P.A. (2024). Network virtualisation utilising blockchain: A review. *Journal of Applied Research in Electrical Engineering*, 3(2), 136-158. doi: 10.22055/jaree.2024.46144.1110.
[27] Wysocki, W., Price, G., Friedman, S., & Conage, A. (2024). *Advanced cyber testing with virtualisation*. doi: 10.4271/2024-01-3893.
[28] Xu, Z., Petrunin, I., & Tsourdos, A. (2021). Dynamic spectrum management with network function virtualisation for UAV communication. *Journal of Intelligent & Robotic Systems*, 101, article number 40. doi: 10.1007/s10846-021-01318-0.
[29] Yan, J., Yang, B., Su, L., He, S., & Dong, N. (2021). Decentralised certificate management for network function virtualisation (NFV) implementation in 5G networks. In J. Xiong, S. Wu, C. Peng & Y. Tian (Eds.), *Mobile multimedia communications* (pp. 81-93). Cham: Springer. doi: 10.1007/978-3-030-89814-4_6.
[30] Yang, K., & Xu, Y. (2024). CNN based resource management for D2D networks with wireless networks virtualisation. In W. Wang, X. Liu, Z. Na & B. Zhang (Eds.), *Communications, signal processing, and systems* (pp. 31-40). Singapore: Springer. doi: 10.1007/978-981-99-7505-1_4.

Віртуалізація та керування мережею: найкращі методи підвищення ефективності

Олександр Берестовенко
Аспірант
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
03056, просп. Берестейський, 37, м. Київ, Україна
https://orcid.org/0000-0003-4887-4674

Анотація. Метою роботи було визначення оптимальних підходів до підвищення ефективності процесів мережевого менеджменту та віртуалізації. Під час дослідження було розглянуто чотири ключові методи: оптимізація розподілу ресурсів за допомогою інтелектуальних алгоритмів, прогнозування навантаження з використанням моделей Machine Learning, динамічне балансування навантаження за допомогою технологій програмно-визначених мереж, а також автоматизоване управління ресурсами на основі політик. Основні результати включали детальний аналіз кожного із запропонованих методів, а саме опис їхніх принципів роботи та етапів впровадження. У дослідженні були представлені схеми, які демонструють архітектуру та механізми функціонування цих методів, а також наведено приклади їх практичного застосування в різних інфраструктурах, таких як хмарні середовища, програмно-визначені мережі та корпоративні дата-центри. Крім того, показано програмні реалізації на мові Python, які дозволили наочно продемонструвати роботу запропонованих підходів. Загалом, результати вказали, що оптимізація розподілу ресурсів забезпечила ефективне використання обчислювальних потужностей у хмарних середовищах, а прогнозування навантаження допомагло заздалегідь адаптувати інфраструктуру до пікових періодів активності. Балансування навантаження на основі програмно-визначених мереж дозволило централізовано керувати трафіком і знижувати затримки, що є критичним для сучасних корпоративних мереж. Автоматизоване управління ресурсами за допомогою політик забезпечило зниження витрат та підтримку стабільності систем шляхом гнучкого реагування на зміну навантаження. В свою чергу, проведене порівняння методів показало, що кожен із методів має свої переваги та обмеження, які потрібно враховувати залежно від специфіки інфраструктури. Отримані результати підтверджують доцільність застосування розглянутих підходів для підвищення продуктивності та стабільності віртуалізованих середовищ і мережевих систем

Ключові слова: розподіл обчислювальних ресурсів; прогнозування навантаження; динамічне балансування; автоматизація процесів; оптимізація трафіку