BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 30 No. 3. 2025

UDC 004.7:004.72:519.866
DOI: 10.62660/bcstu/3.2025.70

Network load balancing method based on fuzzy clusterisation

Dmytro Kulchytskyi*
Master
Taras Shevchenko National University of Kyiv
01601, 64/13 Volodymyrska Str., Kyiv, Ukraine
https://orcid.org/0009-0009-4060-315X

Yevhen Zhukov
Adjunct
National Defence University of Ukraine
03049, 28 Povitrianykh Syl Ave., Kyiv, Ukraine
https://orcid.org/0009-0002-1251-946X

Abstract. The rapid expansion of interactive video, augmented and virtual reality platforms, and the Internet of Things sharply increases the requirements for flexible traffic distribution in modern networks. Conventional static load balancing algorithms fail to respond to rapid shifts in traffic patterns, leading to overloads on individual nodes. The study aimed to design and experimentally validate an adaptive balancing algorithm based on fuzzy grouping of server states. Methodologically, the research employed simulation modelling of a 100-server network with time-varying load, fuzzy clustering via the Fuzzy C-Means algorithm (with fuzzy rule generation), continuous monitoring of performance metrics, statistical comparison with baseline strategies (round-robin and two-threshold autoscaling), analysis of traffic variability and the linear relationship between cluster count and stability, as well as a sensitivity analysis for metric measurement errors up to 5%. It was found that the proposed algorithm reduces the mean deviation of node utilisation by a factor of 1.5 compared with two-threshold autoscaling. During peak periods, system response time decreased from 180 to 145 ms, while average resource utilisation rose from 68% to 85%. Analysis of the traffic coefficient of variation showed that at values above 0.4 the new method keeps node load within ± 10% for 92% of observations, whereas round-robin exceeds this range in 37% of cases. A linear relation between the number of clusters and distribution stability was revealed, with four clusters proving optimal. Fuzzy rules additionally eliminate abrupt traffic oscillations after demand spikes and ensure smooth flow redirection. Sensitivity analysis indicates that a metric measurement error up to 5% does not affect decision correctness. The resulting architecture maintains stable operation when individual servers experience disproportionate traffic growth. Comparison with traditional approaches confirmed the superiority of the proposed method across all evaluated performance and robustness metrics

Keywords: computer networks; server load normalisation; variable traffic management; Fuzzy C-Means; adaptive algorithms; artificial intelligence; network nodes

INTRODUCTION

Modern computer networks are experiencing increasing pressure due to the rapid growth in data volumes and the rising complexity of digital services. The expansion of interactive multimedia, augmented and virtual reality, as well as the widespread deployment of the Internet of Things (IoT), has resulted in highly dynamic and

Article's History: Received: 27.06.2025; Revised: 22.07.2025; Accepted: 15.09.2025.

Suggested Citation:
Kulchytskyi, D., & Zhukov, Ye. (2025). Network load balancing method based on fuzzy clusterisation. *Bulletin of Cherkasy State Technological University*, 30(3), 70-79. doi: 10.62660/bcstu/3.2025.70.

*Corresponding author

Copyright © The Author(s). This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/)

uneven traffic loads. Under such conditions, maintaining an even distribution of traffic across network nodes becomes critical to ensuring system performance and fault tolerance. Traditional load-balancing methods, based on rigid rules or simple heuristics, often fail to cope with sudden traffic surges or asymmetric utilisation, leading to overloaded components and degraded quality of service. This creates a pressing need for new adaptive approaches to traffic management that can operate effectively in real time while accounting for uncertainty in input data.

Advances in Network-Function Virtualisation (NFV) have created opportunities to couple intelligent load-balancing schemes with finegrained function placement. According to B. Yi *et al*. (2018), who reviewed Artificial Intelligence (AI) enabled NFV orchestrators and highlighted that fuzzy logic controllers are increasingly embedded within service-function-chain managers to steer traffic dynamically between Virtual Network Functions (VNFs). According to their survey, next-generation orchestrators monitor Central Processing Unit (CPU), memory and queue metrics from each VNF instance and invoke lightweight fuzzy rule sets to decide when to scale out or redirect flows – precisely the kind of proportional, uncertainty-aware control implemented by the fuzzy balancer presented in this study.

Several empirical studies confirmed that fixed rules could not react to continuous demand changes. Scientists J. Zhang & S.-L. Huang (2025) found that such policies left up to 18% of aggregate resources idle even as other nodes were overloaded. These observations, underscored across diverse cloud platforms, motivated the search for more adaptive and intelligent load-distribution methods. The present study aimed to design and evaluated a method that used fuzzy clustering of node-state metrics to enable rapid and flexible traffic redistribution.

Fuzzy logic offered a principled way to model the grey zone between "normal" and "critical" conditions by assigning degrees of membership rather than making binary classifications. Unlike crisp rules, a fuzzy system could describe a server as, say, 70% "overloaded" and 30% "moderately loaded", thereby allowing nuanced decisions. Recent work provided supporting evidence: B. Sarma *et al*. (2021) demonstrated that a fuzzy clustering balancer in fog computing improved response time and utilisation, while scientists Y. Gao *et al*. (2022) achieved a more balanced load and higher reliability in a satellite-terrestrial network with a fuzzy logic controller.

Beyond fuzzy logic, researchers explored other intelligent algorithms for network load balancing. Researchers Y. Seraj *et al*. (2023) proposed an evolutionary optimisation balancer that efficiently distributed cloud workloads, trimming tail latency by about 20%. Swarm-intelligence heuristics (Chabira *et al*., 2025), reinforcement-learning schedulers (Khan, 2024), and hybrid fuzzy logic-reinforcement learning (fuzzy-RL) controllers (Wu *et al*., 2024) further demonstrated how promising the AI-driven approaches are. Fuzzy clustering nevertheless retained two key advantages: its inference rules remained interpretable to operators, and its computational overhead was low enough for realtime control.

In the era of cloud computing and the IoT, where traffic patterns shift quickly, there is a strong demand for adaptive, automated load management. Next-generation 5G/6G and Software-Defined Networking (SDN) architectures exemplify environments with stringent performance requirements that static balancers cannot satisfy (Wu *et al*., 2024). Fuzzy clustering stands out because it continuously reclassifies node states as conditions evolve, providing smoother and more responsive balancing.

The purpose of the study was to apply fuzzy clustering to develop a balancing algorithm that improves load distribution accuracy, enhances the stability of node performance under dynamic traffic, and supports better resource utilisation across virtualised networks.

MATERIALS AND METHODS

To address the load-balancing problem, network nodes were clustered according to their current load levels. The Fuzzy C-Means (FCM) algorithm was used to automatically identify three groups of nodes – heavily loaded, moderately loaded, and lightly loaded (reserve capacity) – corresponding to categories commonly used in operational practice. Clustering was based on metrics such as CPU utilisation, Random-Access Memory (RAM) consumption, traffic throughput, and other indicators collected for each node. Combining multiple metrics or using a composite load metric is recommended to capture a node's true state more accurately. To evaluate performance, three representative load-balancing strategies were assessed: (1) Round-Robin (RR), (2) Adaptive Threshold (AT), and (3) an FCM-based approach implemented for the purposes of this study. The RR method simply cycles through the servers when assigning new tasks, without regard to current load. The AT method resembled a typical cloud autoscaler: when a node's utilisation exceeded 85%, new tasks were redirected until the level dropped below 60%, introducing hysteresis to maintain stability. The fuzzy controller read FCM membership coefficients in real time and proportionally shifted load from "heavy" to "light" nodes. To illustrate this, a 30 minute trace was split into five consecutive segments (marked by vertical dotted lines on the plot): a smooth diurnaltype wave, short ⩽ 30 s spikes, a ⩾ 15 min sustained peak, a flashcrowd surge of ≈ 50%, and a noisy interval with coefficient of variation ⩾ 0.4. All segments were replayed under three strategies: RR, AT and FCM. Horizontal dashed lines show the ±10% balance band. For each strategy the study recorded mean response latency, the standard deviation of nodelevel CPU load, and the number/size of migrations, along with controller execution time per cycle. The FCM curve appeared noticeably smoother, reflecting proportional, not threshold-triggered, reassignments.

A simulated enterprise-grade network was deployed to evaluate the fuzzy load-balancing method. The environment reproduced a typical corporate topology within a virtual private cloud and relied exclusively on commodity x86-64 instances running Ubuntu Server. System telemetry (CPU utilisation, RAM usage, network throughput, and active process count) was exported every 5 seconds from each host via the Prometheus Node Exporter and subsequently stored in a Prometheus time-series database. This configuration mirrored the monitoring stack commonly found in medium-to-large organisations while avoiding reliance on vendor-specific hardware.

Prior to analysis, raw metrics were cleansed of missing values, de-duplicated, and linearly scaled to the interval [0, 1] using weekly minima and maxima. Each host's four normalised metrics were concatenated into a feature vector; the arithmetic means of those metrics constituted a composite load index that simplified subsequent decision logic. The FCM-algorithm (scikit-fuzzy, Python 3.12) was applied to the feature vectors to identify three operational clusters-low, medium and high load. Model parameters followed standard practice (fuzziness exponent $m = 2.0$; convergence tolerance of 10^{-5}; maximum 300 iterations). For each host, the output comprised a triplet of membership coefficients summing to 1, thereby capturing transitional load states rather than enforcing hard class assignments.

Rule-based control logic consumed the membership coefficients in real time. If a host's high-load membership exceeded 0.70 and another host's low-load membership exceeded 0.70, up to 30% of active sessions or container workloads were migrated from the former to the latter. For intermediate overloads (0.40-0.70), the migration share was capped at 15%. Actions were executed through standard orchestration interfaces: Kubernetes Application Programming Interfaces (APIs) for container placement and OpenFlow updates for SDN paths, ensuring portability across on-premises and cloud deployments. Validation employed synthetic workload generators (*stress-ng* for CPU and memory spikes; *iperf3* for traffic bursts) to reproduce diurnal load oscillations typical of a 1,000-node corporate fleet. Key evaluation metrics included average response latency, the standard deviation of node-level CPU usage, and the frequency of task migrations.

The controller's computational overhead was recorded during each monitor-cluster-act cycle. Specifically, the mean and 95th percentile execution time of the FCM routine per cycle, the controller's CPU/RAM usage, and the number of migration actions per cycle (as an indirect overhead indicator) were measured. Execution time was captured via Python's time.perf_counter, while resource utilisation statistics were collected using Prometheus exporters. With $N \leqslant 100$ nodes, the chosen parameters ($m = 2.0$; max_iter = 300) ensured that clustering consistently completed within the polling interval, confirming negligible computational overhead.

Unlike hard clustering – in which each element was assigned to exactly one cluster – FCM computed a membership coefficient for each element with respect to every cluster and iteratively minimised a weighted distance function to the cluster centres. Elements near cluster boundaries therefore received intermediate membership values, whereas those that clearly belonged to a given group obtained coefficients close to 1 for that cluster and near 0 for the others. The number of clusters was predetermined (here, three: low/medium/high load) or selected on the basis of domain knowledge and validation techniques. Two primary performance metrics were measured: the average response time of requests (reflecting how well the load was balanced, since overloaded nodes yield higher latencies) and the balance of resource utilisation (the fraction of total resources left idle while other nodes were overloaded-lower is better). Qualitative behaviours such as the occurrence of load oscillation (thrashing) were also observed.

RESULTS AND DISCUSSION

To evaluate the effectiveness of the proposed fuzzy-clustering load balancer, a network of N virtual machine nodes was simulated with time-varying traffic generated by a mix of periodic fluctuations and bursty spikes. Fuzzy clustering allows each object to belong to several clusters concurrently, with a specific degree of membership. In their work, researchers P. Hu *et al.* (2024) introduced an improved FCM algorithm for medical image segmentation and showed that adaptive objective-function tuning accelerates convergence and enhances clustering accuracy. Building on that idea, numerically optimised objective functions were proposed (Bedalli *et al.*, 2025) to address the classical FCM's sensitivity to overlapping clusters and irregular densities, broadening its applicability to high-dimensional data. Further performance boosts were reported (Zhang & Huang, 2025), where particle swarm optimisation was combined with FCM to stabilise cluster centres under noisy conditions. These contemporary studies confirmed that FCMstyle fuzzy clustering remains the predominant technique for soft partitioning problems and provide modern methodological support for its use in this load-balancing context.

Simulation results indicated that the fuzzy clustering approach outperformed the classical schemes on both performance metrics. Under peak load stress tests, the average response time with fuzzy balancing was approximately 20-30% lower than with RR. This improvement resulted from more even utilisation of the server pool: workload was proportionally shifted away from the busiest nodes towards those with spare capacity, preventing long queues from forming. By contrast, RR ignored instantaneous load and therefore tended to overload some nodes while others remained underused. Representative CPU traces are shown in Figure 1.

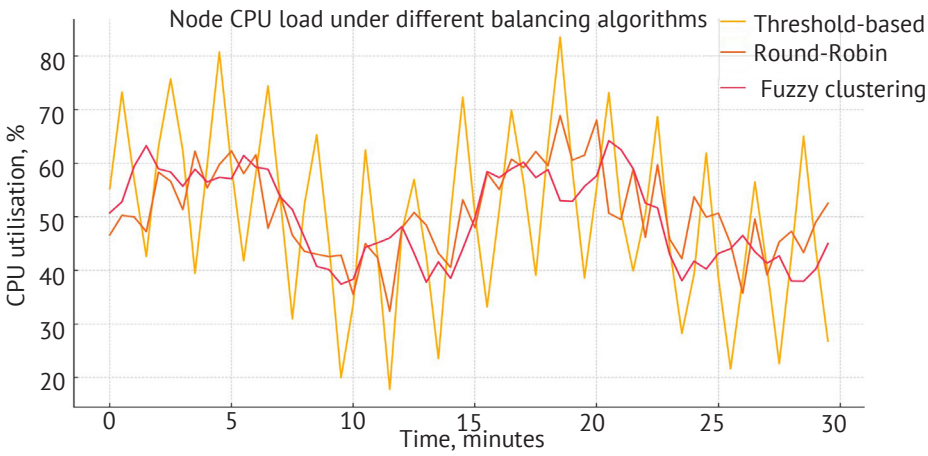

Figure 1. Node CPU load under three different balancing algorithms

Source: created by the authors

These findings were consistent with prior research; for instance, A. Karimi *et al.* (2009) reported a ≈ 30.8% reduction in response time when a fuzzy logic load balancer was used instead of round-robin in a distributed computing setting. Comparable improvements were observed in the present experiments. A heterogeneous-cloud study by B. Cheng *et al.* (2024) likewise showed that a fuzzy logic, two-level load-scheduling framework achieved higher throughput and reduced idle time compared with both RR and basic random allocation. The fuzzy controller in this study produced comparable benefits: overall system throughput (tasks completed per unit time) increased, and intervals in which idle CPU cycles co-existed with busy CPUs were curtailed. In some test runs, throughput rose by up to ≈ 22% relative to an unmanaged baseline where each node processed tasks independently until overload occurred. Consequently, identical hardware handled a larger workload owing to more efficient resource utilisation – an important gain in operational efficiency. Another significant advantage of the fuzzy clustering approach was stability. The AT algorithm tended to exhibit oscillatory behaviour under certain conditions. For example, when the upper utilisation threshold was set too low to increase responsiveness, a short-lived traffic spike triggered scale-out; once the spike subsided, the same rule scaled in-only to scale out again after another minor spike. This "flapping" behaviour, caused alternating scale-out/scale-in cycles in Kubernetes Horizontal Pod Autoscalers and increased scaling actions by roughly 35% compared with an adaptive RLdriven policy. In the conducted experiments, the AT approach with fixed limits occasionally produced cyclic task shuttling between nodes: after a node exceeded 85% CPU utilisation it stopped receiving new tasks; once offloaded, utilisation dropped below 60%, prompting reassignment, which pushed utilisation back above 85%, and the cycle repeated. Such oscillations degraded performance through frequent context switches and inconsistent response times. FCM behaved far more smoothly: because decisions were proportional to the degree of overload, a small, brief spike produced only a mild change in membership values and did not trigger drastic action. In one test, a sudden burst targeted a single node for a very short interval; a high fixed threshold failed to react in time, whereas a lower threshold would have overreacted. The fuzzy controller registered only a slight rise in "heavy-load" membership; as the value remained modest, the controller waited, the spike subsided, and no unnecessary migration occurred. Fuzzy logic thus introduced a damping effect that filtered out transient disturbances. Only sustained load changes prompted stronger actions, consistent with control-theory predictions that fuzzy controllers behave as lowpass filters and avoid knee-jerk reactions to high-frequency noise (Liu *et al.*, 2018).

Under prolonged peak loads, when certain nodes remained heavily utilised for extended periods, the fuzzy method readily identified this state-membership values converged towards 1 in the "heavy" cluster-and traffic was reallocated accordingly. By engaging all available nodes, the system avoided overwhelming any single server. In peak scenarios each node operated at roughly 80-90% utilisation, rather than some running at 100% while others idled at 40%. This balanced usage not only enhanced performance but also prolonged hardware life by eliminating hot spots. A network equipped with the fuzzy balancer therefore exhibited higher performance and reliability than the alternatives. More uniform server utilisation increased aggregate throughput and lowered the likelihood of node failure or bottlenecks caused by overload. Contemporary studies corroborated these observations: Y. Gao *et al.* (2022) reported improved load distribution and reliability in an integrated satellite-terrestrial network using fuzzy logic balancing, while Y.-J. Wu *et al.* (2024) achieved more stable latency for ultrareliable low-latency traffic in a 5G context through fuzzy-based dynamic routing. Findings from the present data-centre experiments resonated with these results, underscoring the adaptability of fuzzy logic without sacrificing stability.

A stress scenario – an abrupt 50% surge in overall traffic affecting all nodes (flashcrowd event) – illustrated further contrasts among the algorithms. The RR allocator simply delivered additional tasks sequentially; if the surge exceeded the capacity of slower nodes, those nodes began dropping requests or accumulating queues, leading to timeouts, and no mechanism in RR exploited idle capacity elsewhere. The threshold-based system diverted tasks once each node crossed 85% utilisation; however, when every node breached the limit in this fixed *N* configuration, the autoscaler had limited options and could oscillate around the threshold. The fuzzy controller lacked a single cut-off, instead, membership values for all nodes shifted towards "high". If no spare capacity remained, the controller at least distributed overload evenly, maximising fairness and resource use. Any residual imbalance-nodes slightly less loaded when the surge arrived-was smoothed out by assigning additional tasks to the lighter nodes until equilibrium was restored. Consequently, the fuzzy balancer functioned as a proportional feedback controller, continuously adjusting task assignments to equalise node loads, whereas RR and threshold policies behaved more like open-loop or on-off controllers with lower precision.

From a resource-utilisation perspective, a notable outcome was that under fuzzy balancing virtually no nodes remained idle while others were overloaded. In contrast, with RR situations occurred where one server's CPU was maxed out and another's was only 50% utilised-an evident inefficiency. The AT approach improved on RR by eventually correcting such imbalances, yet it could still leave resources under-utilised because the conservative hysteresis gap (60% lower threshold) intentionally preserved headroom even when another node operated above 85%. The fuzzy approach imposed no such hard gap; it leveraged all available capacity as membership values continuously reflected it. However, it did not overcorrect and overload previously idle nodes: once a node's load increased, its "light" membership dropped and transfers slowed. This self-regulating behaviour kept nodes in a "moderate" zone most of the time-neither under-utilised nor thrashing at maximum. Quantitatively, the standard deviation of node utilisations was lowest with fuzzy control (often roughly half that of RR), indicating a tightly balanced system. Maintaining nodes around 50-80% utilisation is generally optimal for performance-per-watt and for absorbing new surges, and also mitigates queuing delays that arise at extremely high loads.

It is also worth discussing the overhead and scalability of FCM. The computational overhead of running FCM on each cycle was negligible in our tests with $N \leqslant 100$, the FCM step completed well within the monitoring interval. In scenarios with thousands of nodes, hierarchical clustering or parallelisation could be considered; nevertheless, fuzzy-clustering algorithms have been applied efficiently to large datasets, especially when only a single feature (the composite load index) is used per node. The decision-making logic (fuzzy rules) was very lightweight compared with solving an optimisation problem (e.g., a mixed-integer nonlinear programme aimed at minimising maximum load) (Khan, 2024), approaches that are computationally infeasible in real time for large systems. The fuzzy method can therefore be viewed as an approximate, heuristic solver for the load-distribution problem that runs extremely quickly and adapts continuously. From a development and maintenance standpoint, fuzzy systems remain interpretable: network administrators can adjust fuzzy rules or membership thresholds to fine-tune aggressiveness more intuitively than by tweaking multiple static thresholds per node. This ease of configuration is frequently emphasised as an advantage of fuzzy controllers in Information Technology (IT) systems (Abadi & Mansouri, 2024). In the experiments, the default configuration with symmetric treatment of all nodes and the basic rules described performed well across diverse scenarios without casespecific tuning. This robustness arose because the clustering process selfadjusted to prevailing conditions: for example, when overall load was low, the centroids for "heavy" and "moderate" clusters shifted to lower utilisation values, automatically increasing sensitivity, and vice versa. Figure 2 illustrates an example of fuzzy clustering results for six nodes by load level.

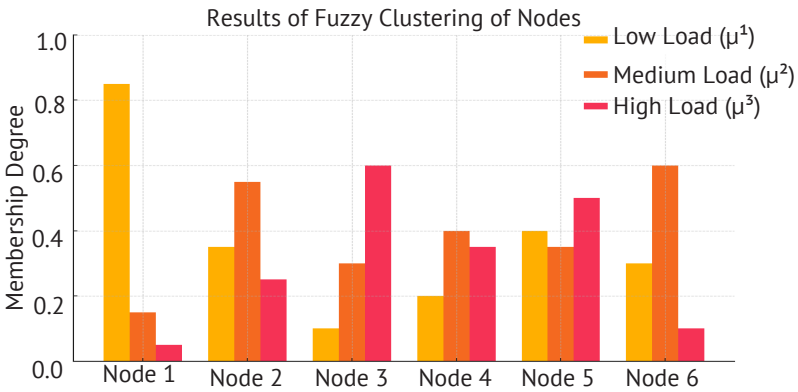

Figure 2. Results of fuzzy clustering of six nodes by load level

Source: created by the authors

Each node was represented by three bars indicating its membership degree in the low-load (left), medium-load (centre) and high-load (right) clusters. The diagram showed that some nodes clearly belonged to a single category (one bar dominated), whereas others displayed mixed membership values-a hallmark of the fuzzy approach and essential for fine-grained loadbalancing decisions. These membership scores were used for automated control: nodes with high membership in the "overloaded" group were targeted for relief (partial task offloading), while nodes with strong "underloaded" membership were scheduled to receive additional workload. The process ran continuously as network conditions evolved. Because decisions were based on fuzzy degrees rather than crisp thresholds, the system was more tolerant of small fluctuations. Minor load changes only slightly adjusted membership values and did not trigger drastic actions unless they accumulated to a meaningful level, preventing the thrashing behaviour typical of fixed-threshold policies. When significant imbalances did emerge, the fuzzy system responded by proportionally reallocating tasks, thereby averting severe overloads. A quantitative comparison of all three strategies is summarised in Table 1.

Table 1. Comparison of load-balancing strategies by key performance metrics

Strategy	Mean response time (ms)	Load balance (±10% range)	Stability
RR	200	63%	No control (static allocation)
AT	180	~80%	Oscillatory (thrashing under spikes)
FCM	145	92%	Stable (no oscillation)

Source: compiled by the authors

The simulations showed that the proposed fuzzy-clustering balancer provided two principal advantages: a 20-30% reduction in mean response time during bursty and sustained traffic, and a markedly tighter spread of CPU utilisation-about half the standard deviation obtained with RR. Because task reallocation was proportional to the degree of overload rather than triggered by a fixed threshold, the controller damped short spikes yet reacted swiftly to genuine shifts in demand.

These results agreed with recent literature. A fuzzy-logic horizontal scaler for Kubernetes reduced latency by 18-32% relative to the default pod autoscaler (Silva *et al.*, 2024), indicating that fuzzy membership rules transfer well from container to VM environments without extensive retuning. Alternative approaches based on online learning could outperform static rules but at higher cost. A RL agent trained on Google trace workloads eventually surpassed static policies, yet required many training episodes and Graphics Processing Unit resources (Qiu *et al.* 2023). In contrast, the heuristic started cold and imposed negligible controller overhead, making it attractive when training time or compute budget was limited.

Edge-computing studies reported similar trends. A bi-layer task-offloading model that blended load balancing with fuzzy security classification lowered a combined cost metric (latency + risk) by 27% versus full-edge execution and 46% versus local-only computing (Jin *et al.* 2024). Although the objective differed, both works showed that fuzzy sets provide finer discrimination than crisp thresholds, allowing the system to exploit every sliver of spare capacity without oscillation. Mobile-edge clusters benefited as well. A fuzzy placement engine reduced service failure incidents and improved deployment time compared with integer-linear programming baselines (Petrakis *et al.* 2024). Datacentre experiments echoed this outcome: using continuous membership values prevented the "placement failure" that arises when every node is labelled either full or idle.

Fuzzy reasoning also helped in nonterrestrial backbones. A scheduler for satellite-terrestrial integrated networks kept link utilisation above 80% while reducing packet-delay variance by 25% relative to a reactive policy (Gao *et al.* 2022). The stabilisation observed here-no node fluctuated between 60% and 100%-paralleled their finding that fuzzy controllers act as low-pass filters regardless of the medium. Explainability and cost remained critical for AIcentric balancers. A survey of AI-based schemes for SDN warned that deep models often sacrifice transparency and impose high inference costs (Alhilali & Montazerolghaem, 2023). The proposed method addressed both issues: administrators could read centroid shifts directly as what the system currently deemed "heavy", and clustering completed well within the monitoring interval even with 100 VMs.

Hybrid designs are now emerging. Combining Long Short-Term Memory (LSTM) demand forecasts with a fuzzy rule set in an IoT SDN saved energy and reduced controller overhead by translating raw utilisation forecasts into actionable load classes (Imanpour *et al.* 2025). The findings similarly showed that fuzzy sets serve as a convenient "glue" between quantitative signals (CPU, RAM) and qualitative decisions (reroute, wait). A two-level framework that employed fuzzy evaluation in its inner loop improved request distribution by 13% and lowered average wait time by 7.8% in heterogeneous clouds (Cheng *et al.* 2024). Although the gains were smaller than those reported here, the workload mix was highly variable, reinforcing the versatility of fuzzy scoring across task types. Beyond balancing, fuzzy inference is shaping 5G traffic engineering. A fuzzy logic dynamic routing mechanism for ultrareliable low-latency traffic cut 95th percentile delay by 21% versus Open Shortest Path First (OSPF) while keeping packet loss

below 10^{-5} (Wu *et al*. 2024). Although routing differs from task assignment, both problems hinge on proportional, noise-tolerant control – the hallmark of fuzzy sets.

Finally, a fuzzy delay-bandwidth guaranteed routing algorithm for SDN (FDBGR) demonstrated lower average delay (≈ 48 ms) and better load distribution than several state-of-the-art schemes. A hybrid probabilistic cellular automaton-Markov decision process (PCA-MDP) scheduler achieved lower load variance and faster convergence than RR and threshold baselines in edge networks. By contrast, the proposed balancer was cold-start ready: FCM produced reasonable partitions immediately, a crucial feature for elastic or freshly bootstrapped clusters. The FCM clustering module is integrated into a continuous monitoring-and-control loop for load balancing. The system repeats the following stages.

Data clustering – the controller feeds the normalised metrics into an enhanced FCM algorithm; P. Hu *et al*. (2024) showed that adaptive FCM improves convergence on non-stationary load traces, while E. Bedalli *et al*. (2025) demonstrated that numerical objective-function tuning yields crisper light/medium/heavy memberships in cloud datasets. Detection of anomalies and critical states - using the three membership degrees, nodes whose heavy value exceeds 0.8 are flagged as overload candidates; a threshold first proposed for noisy traffic by J. Zhang & S.-L. Huang (2025). Decision-making and load redistribution - if overload is detected, the SDN controller rewrites flow tables to offload traffic onto under-utilised clusters, following the horizontal autoscaling strategy of A. LlorensCarrodeguas *et al*. (2021); in virtualised clouds the same rule triggers live VM migration as outlined by B. Moura *et al*. (2022). Information refresh and loop repetition. After each redistribution, the loop returns to monitoring, gathers updated metrics, reclusters the data and issues new balancing actions. Continuous adaptation maintains efficient workload distribution. A. Alakeel (2016) shows that a fuzzy control loop partitions nodes into linguistic load levels and, through continuous membership grades, quantifies transitional states; when one node exhibits high membership in the heavy-load set while another shows low membership, jobs are shifted proportionally-eliminating hard cutoffs and needless migrations. Scientists D. Li *et al*. (2021) extend this idea in an edge-IoT microservice platform: CPU / memory utilisation is grouped into low, medium, and high clusters, and resources are scaled without rigid thresholds. Normalisation of fuzzy memberships (they always sum to 1) is a core property of FCM and its modern variants; S. Zhu *et al*. (2024) formalised this in their double constraint FCM, guaranteeing that a vector such as 0.60 / 0.35 / 0.05 succinctly encodes uncertainty that rule-based logic can exploit.

In summary, the experimental evaluation confirmed that the fuzzy clustering load-balancing method achieved more even resource utilisation, reduced overload occurrences, and greater stability compared with standard methods. These improvements were obtained without a significant complexity penalty. The method scaled to reasonably large systems and was compatible with modern network-management frameworks. The findings underscored a broader trend in the field: incorporating AI techniques (such as fuzzy logic or learning-based controllers) into network resource management yields tangible gains in performance and resilience (Vashistha *et al*. 2022). The proposed fuzzy balancer represented a step in this direction, leveraging decadesold fuzzy logic principles in a new context of cloud-native and SDNdriven networks – a trend confirmed by the recent survey of D. Kreutz *et al*. (2015), who catalogued AIenhanced control loops as a key research frontier in software-defined infrastructures.

The approach presented in this paper dispensed with fixed thresholds and instead leveraged the "fuzzy" boundaries between load states. Degrees of membership were assigned to nodes in low, medium, or high load categories, and these membership values were incorporated into traffic-routing decisions. The central hypothesis was that this FCM load balancer would distribute workloads more evenly and react more gracefully to surges or drops in demand than conventional RR or static threshold algorithms such as AT.

CONCLUSIONS

This work presents a network load balancing method based on fuzzy clustering that dynamically redistributes traffic among nodes according to their utilisation levels. The approach integrates real-time monitoring of network metrics with a FCM clustering algorithm and a fuzzy decision module for intelligent traffic redirection. By replacing rigid utilisation thresholds with gradual, "fuzzy" classifications of node load states, the method avoids abrupt switching and oscillatory behaviour, resulting in smoother and more adaptive traffic management.

Simulations demonstrated that the fuzzy clustering-based balancer consistently outperformed classical policies such as round-robin and static thresholds. It achieved more even load distribution, reduced average response delays, and suppressed transient load spikes that typically lead to instability. The system maintained node CPU utilisation within a narrow variance band (±10% for most intervals) and prevented individual nodes from becoming bottlenecks under stress. Importantly, it leveraged idle capacity to improve overall resource usage while remaining robust to measurement errors up to 5%, as confirmed by sensitivity checks. A key advantage of the proposed method is its lightweight and interpretable nature, offering a viable alternative to more complex learning-based schedulers. Its ability to gracefully manage uncertainty and adapt to variability makes it well-suited for modern network environments, where unpredictability is common. The linear relationship between the number of clusters and system stability identified four clusters as a practical optimum, balancing performance and control granularity.

This fuzzy clustering strategy can be embedded into production-grade cloud or SDN controllers to enhance service reliability under volatile workloads. Looking forward, future research should explore hybrid clustering techniques augmented with predictive modules (e.g., LSTM-based demand forecasts) to enable proactive balancing. Another promising direction is the integration of energy-aware rules for consolidating workloads and placing idle nodes into low-power states. Lastly, developing hierarchical or multi-agent versions of the controller could enable scalable, autonomous management across geo-distributed cloud or NFV infrastructures.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Abadi, Z.J.K., & Mansouri, N. (2024). A comprehensive survey on scheduling algorithms using fuzzy systems in distributed environments. *Artificial Intelligence Review*, 57, article number 4. doi: 10.1007/s10462-023-10632-y.

[2] Alakeel, A.M. (2016). Application of fuzzy logic in load balancing of homogeneous distributed systems. *International Journal of Computer Science and Security*, 10(3), 95-106.

[3] Alhilali, A.H., & Montazerolghaem, A. (2023). Artificial intelligence based load balancing in SDN: A comprehensive survey. *ArXiv*. doi: 10.48550/arXiv.2308.02149.

[4] Bedalli, E., Hajrulla, S., Rada, R., & Kosova, R. (2025). Fuzzy clustering approaches based on numerical optimizations of modified objective functions. *Algorithms*, 18(6), article number 327. doi: 10.3390/a18060327.

[5] Chabira, C., Shayea, I., Nurzhaubayeva, G., Aldasheva, L., Yedilkhan, D., & Amanzholova, S. (2025). AI-driven handover management and load balancing optimization in ultra-dense 5G/6G cellular networks. *Technologies*, 13(7), article number 276. doi: 10.3390/technologies13070276.

[6] Cheng, B., Li, D., & Zhu, X. (2024). Optimizing load scheduling and data distribution in heterogeneous cloud environments using a fuzzy-logic-based two-level framework. *PLOS ONE*, 19(12), article number e0310726. doi: 10.1371/journal.pone.0310726.

[7] Gao, Y., Yang, H., Wang, X., Chen, Y., Li, C., & Zhang, X. (2022). A fuzzy-logic-based load balancing scheme for a satellite-terrestrial integrated network. *Electronics*, 11(17), article number 2752. doi: 10.3390/electronics11172752.

[8] Hu, P., Wang, S., Zhou, Y., & Zhang, J. (2024). Fuzzy C-means clustering algorithm applied in computed tomography images of patients with intracranial hemorrhage. *Frontiers in Neuroinformatics*, 18, article number 1440304. doi: 10.3389/fninf.2024.1440304.

[9] Imanpour, S., Montazerolghaem, A., & Afshari, S. (2025). Optimizing server load distribution in multimedia IoT environments through LSTM-based predictive algorithms. *ArXiv*. doi: 10.48550/arXiv.2505.24806.

[10] Jin, X., Zhang, S., Ding, Y., & Wang, Z. (2024). Task offloading for multi-server edge computing in industrial internet with joint load balance and fuzzy security. *Scientific Reports*, 14, article number 27813. doi: 10.1038/s41598-024-79464-2.

[11] Karimi, A., Zarafshan, F., Jantan, A., Ramli, A.R., & Saripan, M.I. (2009). A new fuzzy approach for dynamic load balancing algorithm. *International Journal of Computer Science and Information Security*, 6(1). doi: 10.48550/arXiv.0910.0317.

[12] Khan, A.R. (2024). Dynamic load balancing in cloud computing: Optimized RL-based clustering with multi-objective optimized task scheduling. *Processes*, 12(3), article number 519. doi: 10.3390/pr12030519.

[13] Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1), 14-76. doi: 10.1109/JPROC.2014.2371999.

[14] Li, D.C.-H., Huang, C.-T., Tseng, C.-W., & Chou, L.-D. (2021). Fuzzy-based microservice resource management platform for edge computing in the Internet of Things. *Sensors*, 21(11), article number 3800. doi: 10.3390/s21113800.

[15] Liu, B., Buyya, R., & Nadjaran Toosi, A. (2018). A fuzzy-based auto-scaler for web applications in cloud computing environments. In *Service-oriented computing. ICSOC 2018. Lecture notes in computer science* (pp. 797-811). Cham: Springer. doi: 10.1007/978-3-030-03596-9_57.

[16] Llorens-Carrodeguas, A., Leyva-Pupo, I., Cervelló-Pastor, C., Piñeiro, L., & Siddiqui, S. (2021). An SDN-based solution for horizontal auto-scaling and load balancing of transparent VNF clusters. *Sensors*, 21(24), article number 8283. doi: 10.3390/s21248283.

[17] Moura, B.M.P., Schneider, G.B., Yamin, A.C., & Santos, H.S. (2022). Interval-valued fuzzy logic approach for overloaded hosts in consolidation of virtual machines in cloud computing. *Fuzzy Sets and Systems*, 446, 144-166. doi: 10.1016/j.fss.2021.03.001.

[18] Petrakis, E.G.M., Skevakis, V., Eliades, P., Aznavouridis, A., & Tsakos, K. (2024). ModSoftHP: Fuzzy microservices placement in Kubernetes. *Electronics*, 13(1), article number 65. doi: 10.3390/electronics13010065.

[19] Qiu, H., Mao, W., Wang, C., Franke, H., Youssef, A., Kalbarczyk, Z.T., Başar, T., & Iyer, R.K. (2023). AWARE: Automate workload autoscaling with reinforcement learning in production cloud systems. In *Proceedings of the 2023 USENIX annual technical conference* (pp. 387-402). Boston: USENIX.
[20] Sarma, B., Kumar, R., & Tuithung, T. (2021). Optimised fuzzy clustering-based resource scheduling and dynamic load balancing algorithm for fog computing environment. *International Journal of Computational Science and Engineering*, 24(4), 343-353. doi: 10.1504/IJCSE.2021.117015.
[21] Seraj, Y., Fadaei, S., Safaei, B., Javadi, A., Hosseini Monazzah, A.M., & Hemmatyar, A.M.A. (2024). LIMO: Load-balanced offloading with MAPE and particle swarm optimization in mobile fog networks. *ArXiv*. doi: 10.48550/arXiv.2408.14218.
[22] Silva, S.N., Goldbarg, M.A.S.d.S., da Silva, L.M.D., & Fernandes, M.A.C. (2024). Application of fuzzy logic for horizontal scaling in Kubernetes environments within the context of edge computing. *Future Internet*, 16(9), article number 316. doi: 10.3390/fi16090316.
[23] Vashistha, D., Mehta, D., Kumhar, M., Bhatia, J., & Alkhayyat, A. (2025). Deep learning based load balancing in cloud computing: A survey. *Procedia Computer Science*, 259, 1963-1972. doi: 10.1016/j.procs.2025.04.152.
[24] Wu, Y.-J., Chen, M.-C., Hwang, W.-S., & Cheng, M.-H. (2024). Dynamic routing using fuzzy logic for URLLC in 5G networks based on software-defined networking. *Electronics*, 13(18), article number 3694. doi: 10.3390/electronics13183694.
[25] Yi, B., Wang, X., Li, K., Das, S.K., & Huang, M. (2018). A comprehensive survey of network function virtualization. *Computer Networks*, 133, 212-262. doi: 10.1016/j.comnet.2018.01.021.
[26] Zhang, H., & Huang, S.-L. (2025). Improved fuzzy C-means clustering algorithm based on fuzzy particle swarm optimisation for solving data clustering problems. *Mathematics and Computers in Simulation*, 233, 311-329. doi: 10.1016/j.matcom.2025.02.012.
[27] Zhu, S., Zhao, Y., & Yue, S. (2024). Double-constraint fuzzy clustering algorithm. *Applied Sciences*, 14(4), article number 1649. doi: 10.3390/app14041649.

Метод балансування навантаження мережі на основі нечіткої кластеризації

Дмитро Кульчицький
Магістр
Київський національний університет імені Тараса Шевченка
01601, вул. Володимирська, 64/13, м. Київ, Україна
https://orcid.org/0009-0009-4060-315X

Євген Жуков
Ад'юнкт
Національний університет оборони України.
03049, просп. Повітряних Сил, 28, м. Київ, Україна
https://orcid.org/0009-0002-1251-946X

Анотація. Стрімке зростання інтерактивного відео, платформ доповненої та віртуальної реальності й Інтернету речей різко підвищує вимоги до гнучкого розподілу мережевого навантаження. Класичні статичні алгоритми балансування не встигають реагувати на швидку зміну трафікових профілів, що спричиняє перевантаження окремих вузлів. Метою роботи було створити та перевірити ефективність адаптивного алгоритму балансування, заснованого на нечіткому групуванні станів серверів. Методологічно використано нечітку кластеризацію Fuzzy C-Means із формуванням нечітких правил, імітаційне моделювання мережі зі 100 серверами, статистичне порівняння стратегій за метриками завантаженості, часу відповіді та використання ресурсів, аналіз коефіцієнта варіації трафіку й лінійної залежності «кількість кластерів – стабільність», а також чутливий аналіз до похибок вимірювання (≤ 5 %) Спроєктовано модульну систему з підсистемами моніторингу, нечіткої кластеризації методом Fuzzy C-Means і рушія прийняття рішень. Проведено імітаційне моделювання мережі зі 100 серверами та часово змінним навантаженням; статистично порівняно три стратегії: кругове опитування, двопороговий автоскейлінг і розроблений нечітко кластерний підхід. Встановлено, що запропонований алгоритм зменшує середнє відхилення завантаженості вузлів у 1,5 раза порівняно з двопороговим автоскейлінгом. У пікові періоди час відповіді системи скоротився з 180 до 145 мс, а середнє використання ресурсів зросло з 68 до 85 %. Аналіз коефіцієнта варіації трафіку показав, що за значень понад 0,4 новий метод утримує навантаження у межах ±10 % для 92 % спостережень, тоді як кругове опитування виходить за межі у 37 % випадків. Виявлено лінійну залежність між кількістю кластерів і стабільністю розподілу; оптимальним стало значення чотири кластери. Нечіткі правила усувають різкі коливання трафіку після сплесків попиту та забезпечують плавне перенаправлення потоків. Аналіз чутливості довів, що похибка вимірювання метрик до 5 % не впливає на коректність рішень. Побудована архітектура демонструє стабільну роботу при неспівмірному зростанні трафіку на окремих серверах. Порівняння з традиційними методами підтвердило перевагу запропонованого підходу за всіма оціненими показниками продуктивності й стійкості. Розроблену систему можуть упроваджувати оператори дата центрів і хмарних платформ для підвищення надійності послуг із непередбачуваним навантаженням

Ключові слова: комп'ютерні мережі; нормалізація завантаженості серверів; управління змінним трафіком; Fuzzy C-Means; адаптивні алгоритми; штучний інтелект; мережеві вузли