Leveraging Community-based Approaches for Enhancing Resource Allocation in Fog Computing Environment

Alasef M. Ghalwah
Department of Information Networks, College of Information Technology, University of Babylon, Babylon, Iraq
alasefmohammedr.net@student.uobabylon.edu.iq (corresponding author)

Ghaidaa A. Al-Sultany
Department of Information Networks, College of Information Technology, University of Babylon, Iraq | College of Information Technology Engineering, Alzahraa University for Women, Karbala, Iraq
Ghaidaa.almulla@alzahraa.edu.iq

Received: 8 October 2024 | Revised: 8 November 2024 | Accepted: 3 December 2024

 DOI: https://doi.org/10.48084/etasr.9206

ABSTRACT

Efficient resource allocation in fog computing environments is essential to address the increasing demand for high-performance and adaptable network services. Traditional methods lack granular differentiation based on traffic characteristics often resulting in suboptimal bandwidth utilization and elevated latency. To enhance network efficiency, this study applies a community-based resource allocation approach leveraging the Louvain algorithm to dynamically cluster network nodes with similar traffic demands. By forming communities based on bandwidth and latency needs, this approach enables a targeted resource distribution, aligning each community with optimized pathways that address specific requirements. The results indicate notable performance gains, including a 14% increase in bandwidth utilization affecting the download and a reduction in latency by an average of 23% for time-sensitive applications. These improvements highlight the effectiveness of the proposed approach in managing diverse network demands, improving data flow stability, and enhancing the overall performance of fog computing infrastructures. These findings underscore the potential for community-based resource allocation to support scalable, adaptable, and secure resource management, positioning it as a viable solution to meet the complex needs of IoT and other distributed network systems.

Keywords-quality of service; community-based algorithms; Louvain method; traffic characteristics; throughput; network stability

I. INTRODUCTION

In the domain of Local Area Networks (LANs), fog and cloud computing have substantially improved their performance, offering a very scalable and elastically centralized system [1-2]. The main extension of the cloud architecture is fog computing, which brings some of the processing near the end user, uplifting some of the overhead from the cloud. Thus, to reduce latency and enhance local data availability, these benefits are based on sublime data management and processing at fog nodes, as well as computing power and efficient data management in geographically broad locations [3-5]. One of the most frequent challenges in LANs is maintaining network performance and availability. To address such challenges, resource allocation methods are implemented. The nature of the network varies depending on the traffic generated by the nodes, making it very difficult to use a single method. Thus, there is a need for sophisticated load balancing that would employ advanced algorithms to distribute incoming requests and computational loads as most suitable to the network as possible. In this sense, no service is supposed to become a weak link and cause a single point of failure in the network, thus improving the robustness and responsiveness of the overall system [6-8].

The most recent studies have proposed adaptive approaches using predictive analytics and machine learning. However, these methods can improve performance but may also increase overhead. On the other hand, these methods are dynamic and adapt to changing conditions to optimize resource utilization. Community detection with the Louvain method to balance loads could make a real difference in how resources are managed in LANs and ISPs, allowing for more accurate resource allocation. Context-based resource distribution takes advantage of the similarity between nodes and manages them

in communities to ensure adequate performance gain without massive overhead on the network crippling throughput [9].

This study proposes a method that combines Louvain and traffic redirection methods to improve network stability and increase its throughput by creating communities of nodes based on similar characteristics and interests in network resources. While building the network tree, an entry is stored for the best requirements matching the community and redirecting the community traffic to its most suitable path. IoT devices that rely on low latencies can benefit from this framework as they are forwarded to the lowest latency paths. For the nodes demanding higher bandwidth, their paths are uncontested by the other nodes, as they are in specific communities directed to the highest bandwidth paths. Furthermore, the communities are reshaped and recreated once conditions are met to ensure sufficient wastage in the network resources and the community, allowing for better resource distribution among all the nodes.

II. RELATED WORKS

In [10], the resource allocation challenges in fog computing environments were examined, focusing mainly on latency-sensitive applications. This systematic mapping study reviewed existing strategies and highlighted gaps for future research, notably the need for efficient allocation mechanisms that can dynamically adapt to changing network conditions and application demands. In [11], a review of resource allocation strategies within fog and cloud environments was presented. This study highlighted the challenges posed by the dynamic nature of these networks and the heterogeneity of devices involved. Various strategies to enhance resource allocation were explored, and the influence of network characteristics on performance metrics, such as energy consumption, latency, and cost, were discussed. The study in [12] delved into optimizing Kubernetes scheduling algorithms. Existing approaches were classified into four categories: generic scheduling, multi-objective optimization-based scheduling, AI-focused scheduling, and autoscaling-enabled scheduling. The review identified significant gaps and proposed directions for future research, emphasizing the need for improved scheduling efficiency and reduced resource consumption in cloud environments.

The study in [13] focused on reliable resource allocation and management for IoT transportation using fog computing. Strategies considering latency and energy efficiency were proposed, which are critical to ensure robust performance in dynamic and decentralized IoT environments. The findings showed that the proposed strategies significantly reduced latency and energy consumption, enhancing the overall sustainability and efficiency of IoT systems. In [14-15], optimization of computation offloading and resource allocation in cloud-fog cooperative networks was explored. Novel simulated annealing algorithms were introduced to optimize these processes, demonstrating reduced system costs and improved performance, marking a significant step forward in cloud-fog resource management.

III. PROPOSED METHOD

The proposed method uses the Louvain algorithm to efficiently create communities from the network nodes. This method is the most suitable option for the diverse nature of the network and the rapid changes in its traffic. This approach addresses the challenges of heterogeneous devices that require varying bandwidth and latency levels. The framework of the proposed method consists of several phases in which the system can perform specific tasks.

The Louvain method is a community detection algorithm. Its primary function is to extract the communities of a given network based on their similar interests. It operates in two primary phases: modularity optimization and community aggregation [16].

- Modularity Optimization: Each of the nodes is within a community by itself. In the local moving phase, the algorithm has nodes reassigned to neighboring communities to increase modularity, which quantifies the links within the communities against the number of links between communities. The nodes are then moved iteratively to neighboring communities so that movement increases the total modularity of the network. The convergence phase is in a loop until no modularity improvement can occur.
- Community Aggregation: All previously generated communities are aggregated into supernodes, forming a new, smaller network [18-19]. The Louvain algorithm is repeated for this new network and the same operation will happen again. This process repeats without stopping, while the network size is reduced each time it is done until maximum modularity cannot be achieved. Communities reach maximum stability and final identification.

By clustering nodes based on their interaction patterns, other predefined packet elements, and traffic similarities, the Louvain method will provide the required insights to optimize resource allocation and improve network performance. A community-based approach enables the framework for more specific resource allocation, as each community can be managed and redirected according to its specific requirements [20].

The BestPath method describes the work of multiple functions serving a single point in this framework. This function is set to record the network paths and sort them in a hash table to make them accessible to the framework for further determining the most suitable path. This method involves the following steps.

A. Route Initialization

- Traffic Monitoring: In this step, the state of the method monitors the state of the network as it starts to create traffic. Initial routes are established based on the nodes' requested services and interactions.
- Hash Table Creation: The collected routes are stored in this table, which will be accessible later and used to redirect traffic suitable for the community.

B. Path Sorting

- Bandwidth and Latency: Paths are sorted in the hash table based on two key metrics: bandwidth (in descending order) and latency (in ascending order). This sorting ensures low lookup times and reasonable allocation if more than one community requires similar services. While keeping paths sorted, each path is dedicated to a specific community to ensure that not every path will be assigned to a single individual community to eliminate path exhaustion and reduce path congestion.
- Lookup Optimization: This sorting allows for lower latencies in searching suitable paths, reducing the framework's process time and the node's overall overhead.

C. Path Selection

- Optimal Path Determination: As the nodes start to generate traffic, the most suitable path is chosen to redirect the community traffic to the chosen path.
- Dynamic Adaptation: No network can remain in the same state for long, and the paths are variable. In addition to other challenges that can change their performance, the table continues to update to match the network state.
- Employing the BestPath method, the network state is continually monitored, and the table is constantly updated to keep track of the paths available in the tree and ensure that path selection works on updated paths. Paths that have been assigned to a community are removed from the table to ensure efficient use of system memory and reduce resource usage.

1) Phase 1: Network Cold-State

In this phase, the network is initialized, starting with zero traffic generation. This is a common phenomenon in networks, and for the Louvain method, the network needs more resources to determine the communities. On the other hand, more than the traffic must be considered a viable entry into the hash table. This name implies that this phase occurs only at the very first start of the network.

2) Phase 2: Segmentation

In the second phase, the nodes generate traffic and send requests. At this point, sufficient data is collected to establish a baseline on which the algorithm can work. The Louvain method starts the respective phases and applies them to the nodes to extract the communities inside the network. It starts by suggesting a single community for each node and then comparing the nodes to each other and measuring the distance between them. These steps repeat until maximum modularity is achieved. After each similar cluster is identified, the segmentation ensures that each node is assigned to a community with limited members to ensure that the community's resources are maintained due to the number of members in that community.

3) Phase 3: BestPath Method

In the third phase, the communities have reached their final shape and limits, or the network nodes need to be increased to a new community. All the routes are now recognized and stored in the hash table. As they are stored, sorting is applied to the stored paths, resulting in faster access to these routes. Then, the communities are assigned to paths that suit their requests. The first community takes the first path in the table matching the requested services, i.e., the demanding community is dedicated to the first path/entry in the bandwidth-sorted table, and the same method applies to the latency communities.

4) Phase 4: Community Reshaping

The fourth phase involves the reshaping of existing communities. Nodes on the network can change traffic and their requests. Thus, a function was created to set a timer for the lifespan of each community in the network. As time itself is not a good indicator, a request limit is added to this function. This function rebuilds communities once one or all conditions are met to prevent resource waste and eliminate idle nodes in communities. After various tests of different values, a 200 s timer and a 1300 packet hit/request limit were set as default primary conditions. These values can be adjusted to suit other networks. The size of the testing community consisted of 5 nodes maximum. This phase occurs periodically in intervals of 200 s.

5) Phase 5: Second Best Community Formation

Second communities are created to avoid path monopolization and introduce fair resource allocation to all involved nodes. A second community is created when the primary community reaches its member limit. These communities are redirected to the second entry in the table, the third is matched with the third entry, and so on, reoccurring as a pattern to eliminate path exhaustion.

The main arguments for deciding the communities are intrinsic features in every packet that can help identify the community members and reduce the Louvain method's operational time. Five key features are as follows:

- Destination IP: This field is essential because it is one of the very first signs of similarity between nodes. Nodes sharing the same destination IP are similar in that they require the same network resources and use the same path.
- Maximum Transmission Unit: This feature is critical in deciding whether the node requires heavy bandwidth or low latencies. It serves as a suitable initial indicator for the packet size and future requests demanded by the node. Typically, IoT nodes use lower MTU sizes.
- Packet Rate: Using a counter to determine the number of packets sent by individual nodes generates more significant estimates of the total number for the community and how long it will remain in its original shape. Nodes that generate a more significant number of packets enter rapidly reshaping communities.
- Port Number: Some services use a different port number, allowing for clear state-reading to similar nodes. Nodes using the same port number possibly indicates that they request the same services. Thus sharing the same path is viable.

- Quality of Service: The level of the services requested by the nodes indicates the services required and the resources demanded by the node.

After all nodes have reached their final shape, the system keeps the traffic directed on the paths provided by BestPath until there is a change in the path state, such as losing connection or failure in the path, or until the community reaches its lifespan limit. In case of any of these events, the paths and the communities are rebuilt to ensure minimum outdated routing choices that are no longer optimal.

Algorithm 1 describes the steps of the proposed framework.

```
Algorithm 1: The proposed framework
1.  Input: initialization parameters
2.  Output:
3.  while (ColdState is True) do
      for each node in network do
        generate initial traffic
      End for
      if (traffic is detected) then
        ColdState = False
      End if
    End while
4.  Initialize communities
    For each node in network do
      assign node to initial community
    End for
5.  Repeat process
      For each node do
        BestCommunity =
        find_best_community_for_node
        if (modularity increases) then
          move node to BestCommunity
        End if
      End for
6.  Aggregate communities until
    modularity stabilizes
7.  Initialize hash table for routes
    for each request in network do
      source = request.source
      destination = request.destination
      route = find best route(source,
      destination)
      if (route exists) then
        send data (request, route)
      else
        new_route = create new route(
        source, destination)
        add new_route to the hash table
      End if
    End for
8.  Assign traffic_threshold = 300
    if (network traffic >
    traffic_threshold) then
      for each community do
        if (community traffic >
        traffic_threshold)
          disband community
          for each node in community do
            assign node to new community
            remove idle nodes
          End for
        End if
      End for
    End if
9.  Community classification
    for each community do
      if (community reaches member limit)
        second_community =
        create_second_community
        second_path =
        find_second_best_path(community)
        for each node in community do
          if (node needs second community)
            move node to second_community
            use second_path for data
            transmission
          End if
        End for
      End if
    End for
9.  Finish
```

IV. SIMULATION AND RESULTS

The main metrics of the system are the latency, download time, and CPU utilization.

A. Simulation Setup

The simulation environment involved a diverse network with 21 nodes actively sending requests to various services. Nodes were classified and formed based on their specific bandwidth and latency requirements, reflecting real-world scenarios and network behavior involving heterogeneous and homogeneous devices. The Louvain method was applied to extract communities within the network and provide insight.

B. Execution

The simulation progressed through all the framework's phases. In network initialization, the nodes generate initial traffic after the cold state with no predetermined paths or communities. In community formation, the Louvain method is used to identify and extract communities based on emerging traffic patterns. The implementation of the BestPath method involves storing and sorting paths in a hash table based on bandwidth and latency metrics, allowing for efficient route selection.

The first simulation case demonstrated improvements in the requested metrics, bandwidth, and latency, validating the effectiveness of the Louvain method. By redirecting traffic through optimal paths within communities benefiting mainly from lower latencies, the network achieved a 14% decrease in average latency and nearly 23% lower latency per requested community compared to a network without the proposed framework.

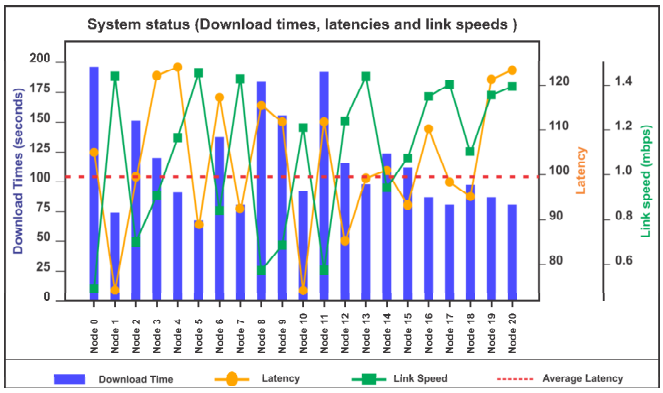

Fig. 1. The trend of higher download times for the low-latency nodes.

The T_d for a single node was calculated using $T_d = \frac{S}{B}$, where S is the size of the data to be downloaded in bits and B is the bandwidth of the path in bps. For multiple nodes, the total download time $T_{d_{\text{total}}}$ was calculated by summing the individual download times of all nodes:

$$T_{d_{\text{total}}} = \sum_{i=1}^{n} \frac{S_i}{B_i}$$

where S_i is the data size, B_i is the bandwidth for node i, and n is the total number of nodes.

This test procedure considered a scenario with three nodes downloading files of sizes 10, 20, and 30 MB over paths with bandwidths of 100, 150, and 200 Mbps, respectively. The calculated download times were 0.8 s, 1.067 s, and 1.2 s, respectively. The total download time for these nodes was 3.067 s.

Similarly, the latency difference ΔL between the two paths was calculated using:

$$\Delta L = L_1 - L_2$$

where L_1 is the latency of the first path and L_2 is the latency of the second path, both in ms. For multiple nodes, the total latency difference ΔL_{total} was calculated by summing the individual latency differences of all nodes:

$$\Delta L_{\text{total}} = \sum_{i=1}^{n} (L_{1i} - L_{2i})$$

where L_{1i} is the initial latency, and L_{2i} is the reduced latency for the node.

In the test results, three nodes with base latencies of 50, 60, and 70 ms were observed, and their latencies were reduced by 25, 30, and 35 ms, respectively, resulting in a total latency for all nodes averaging 90 ms.

Based on the test calculations, the results show improvements in both download times and latency reductions. These results were achieved using the Louvain and BestPath methods, demonstrating the enhanced performance and efficiency of the proposed approach in real-world network environments.

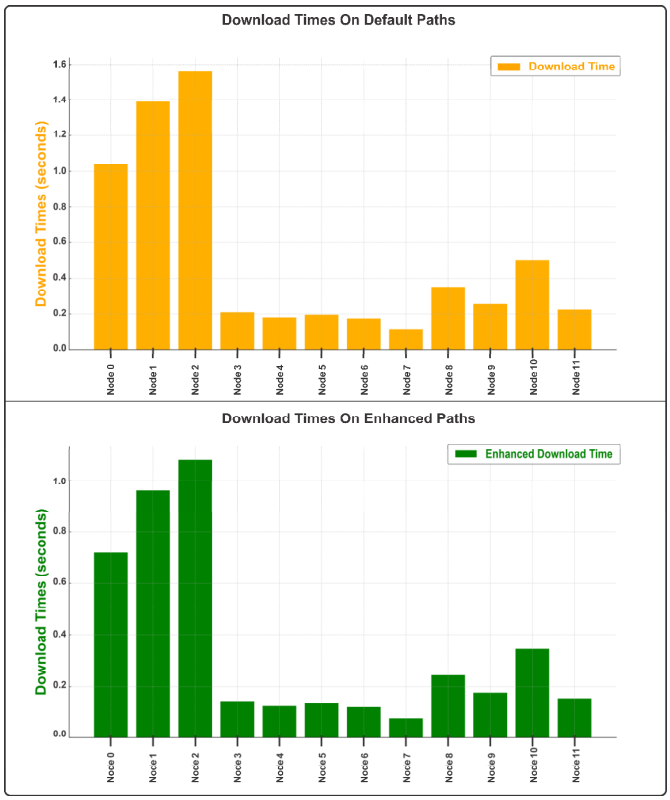

Fig. 2. Bandwidth differences between base and BestPath.

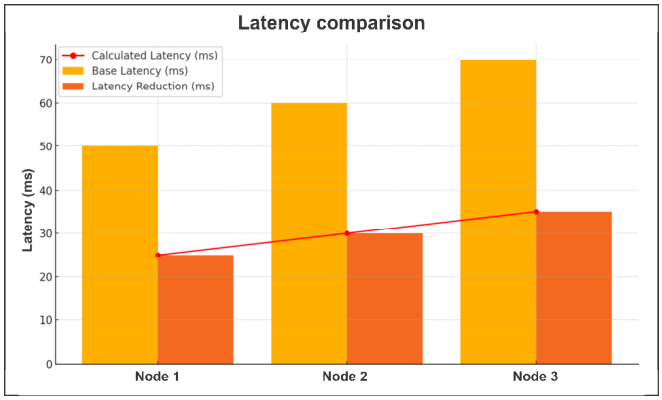

Fig. 3. Latency comparison between base and BestPath.

In the second simulation case, the latency results shown in the table below indicate higher improvements to the average latency of the nodes after redirecting their community traffic. Table I shows the evaluation metrics of the second case study of the proposed system.

In addition, the proposed system was compared with [21], which proposed a resource representation and allocation scheme for edge computing to meet the needs of latency-sensitive applications. By employing a Lyapunov optimization framework, a dynamic resource allocation method was developed, which improved system performance in terms of resource utilization and latency. Simulation results showed that the proposed method outperformed other benchmark methods

in latency reduction and resource consumption optimization. Table II shows the proposed system's specifications compared with related works on computational and latency metrics.

TABLE I. EVALUATION METRICS OF SECOND CASE RESULTS

Node ID	Community	Base path Latency (ms)	Redirected paths Latency (ms)	Latency Improvement (%)
Node 7	C 1	210	96	54.30%
Node 3	C 1	172	90	47.70%
Node 1	C 1	125	66	47.20%
Node 5	C 1	152	74	51.30%
Node 2	C 1	198	94	52.50%
Node 8	C 2	194	88	54.60%
Node 6	C2	185	85	54.10%
Node 4	C 2	240	104	56.70%
Node 11	C 2	138	73	47.10%
Node 10	C 2	163	78	52.10%
Node 14	C3	193	86	55.40%
Node 9	C 3	232	108	53.40%
Node 13	C 3	302	128	57.60%
Node 12	C3	255	111	56.50%

TABLE II. COMPARISON OF SYSTEM SPECIFICATIONS

Resource Provisioning in Edge Computing for Latency Sensitive Applications [21]	
Metric	Values
Latency before enhancement (MGRA)	250 ms
Latency after enhancement (LRR)	150 ms
Percentage of latency reduction	40% latency reduction
CPU utilization before enhancement	High (100% under certain loads)
CPU utilization after enhancement	82%-89% CPU utilization
CPU specifications	2.5 GHz processing power
Proposed system	
Metric	Values
Latency before enhancement	147 ms
Latency after enhancement (BP)	63 ms
Percentage of latency reduction	57.11% latency reduction
CPU utilization before enhancement	Medium-High
CPU utilization after enhancement	58%-72% CPU utilization
CPU specifications	2.5 GHz Intel core I5 10400f

Table III shows the results for the third simulation case, where the improvement in download times (bandwidth utilization) was recorded on 40 nodes. The average latency for base paths across all nodes was 223.0 ms, while BestPath showed an improved average latency of 113.7 ms, resulting in an overall average latency improvement of 49.0%. Considering download times, base paths had an average of 27.5 s compared to BestPath's faster average of 25.9 s, providing an average download improvement of 5.8%. These improvements reflect the reduction in latency and modest gains in download speed when using the BestPath method.

V. CONCLUSIONS

Optimizing network performance is essential for ensuring efficient and effective data transmission. Different methods yield varying results. This study employed the Louvain method to create communities within the network. These communities redirect their traffic to the most optimal paths, sharing them within the community. This approach enhances the user experience and improves QoS by dividing the network load into specific categories. This categorization maximizes the benefits gained from each path, facilitating smooth network operation. As a result, network performance increases, and each path in the network tree is used efficiently.

TABLE III. EVALUATION METRICS FOR THE THIRD SIMULATION CASE

Node ID	Community	Base paths latency (ms)	Redirected paths latency (ms)	Latency improvement (%)	Base download time (s)	Redirected download times (s)	Download improvement (%)
Node 27	C1	290	130	55.20%	32	30.5	4.70%
Node 38	C1	280	250	10.70%	38	37	2.60%
Node 20	C1	150	77	48.70%	20	19	5.00%
Node 35	C1	250	230	8.00%	33	32	3.00%
Node 2	C1	215	105	51.20%	25	23	8.00%
Node 13	C2	270	118	56.30%	29	27	6.90%
Node 25	C2	240	110	54.20%	28	26.5	5.40%
Node 11	C2	155	81	47.70%	21	19.5	7.10%
Node 24	C2	190	88	53.70%	24	22	8.30%
Node 9	C2	160	75	53.10%	19	17.5	7.90%
Node 3	C2	150	71	52.70%	22	20	9.10%
Node 39	C3	290	270	6.90%	40	39	2.50%
Node 14	C3	210	97	53.80%	24	21.5	10.40%
Node 28	C3	210	100	52.40%	25	23	8.00%
Node 23	C3	280	122	56.40%	30	28	6.70%
Node 18	C3	245	120	51.00%	28	26.5	5.40%
Node 8	C3	210	95	54.80%	24	21.5	10.40%
Node 37	C4	265	245	7.50%	36	35.5	1.40%
Node 5	C4	190	90	52.60%	18	16.5	8.30%
Node 7	C4	240	110	54.20%	28	26	7.10%
Node 31	C4	200	180	10.00%	28	27.5	1.80%
Node 12	C4	230	104	54.80%	27	25	7.40%
Node 6	C4	178	85	52.20%	16	15	6.30%
Node 10	C5	170	90	47.10%	20	18.5	7.50%
Node 40	C5	270	245	9.30%	34	33	2.90%
Node 26	C5	250	112	55.20%	29	27	6.90%
Node 32	C5	220	200	9.10%	30	29.5	1.70%
Node 15	C5	200	94	53.00%	23	21	8.70%
Node 17	C5	225	105	53.30%	25	23.5	6.00%
Node 16	C6	190	86	54.70%	22	20.5	6.80%
Node 34	C6	240	220	8.30%	32	31	3.10%
Node 1	C6	130	67	48.50%	20	18.5	7.50%
Node 4	C6	225	98	56.40%	30	27	10.00%
Node 36	C6	210	190	9.50%	29	28.5	1.70%
Node 22	C7	310	140	54.80%	33	30	9.10%
Node 30	C7	220	110	50.00%	26	24.5	5.80%
Node 33	C7	260	235	9.60%	35	34	2.90%
Node 29	C7	225	115	48.90%	27	25	7.40%
Node 21	C7	275	120	56.40%	31	28.5	8.10%

The adaptive nature of the network allows for specific adjustments to be made in all communities, leading to noticeable performance gains across multiple network applications. Nodes that benefit primarily from low-latency paths, such as IoT devices, are redirected to the most suitable paths. Conversely, nodes that require high download and upload speeds are routed to the highest bandwidth paths in the hash table. The proposed method ensures optimal allocation of network resources while maintaining a balanced and efficient network environment for all nodes. Applying specific adjustments to the communities, the Louvain method implemented in the proposed system demonstrates a clear

improvement in network performance, reducing queue times for each node as traffic is redirected to separate optimal paths. In general, implementing the Louvain method for community-based load balancing provides a robust framework for enhancing network performance, making it a viable solution to address the dynamic demands of modern networks.

REFERENCES

[1] T. Salehnia *et al.*, "An optimal task scheduling method in IoT-Fog-Cloud network using multi-objective moth-flame algorithm," *Multimedia Tools and Applications*, vol. 83, no. 12, pp. 34351–34372, Apr. 2024, https://doi.org/10.1007/s11042-023-16971-w.

[2] G. Goel and A. K. Chaturvedi, "Multi-Objective Load-balancing Strategy for Fog-driven Patient-Centric Smart Healthcare System in a Smart City," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 16011–16019, Aug. 2024, https://doi.org/10.48084/etasr.7749.

[3] M. A. Alshahrani, A. A. Qidan, T. E. H. El-Gorashi, and J. M. H. Elmirghani, "Energy Efficient Service Placement for IoT Networks," in *2024 24th International Conference on Transparent Optical Networks (ICTON)*, Bari, Italy, Jul. 2024, pp. 1–5, https://doi.org/10.1109/ICTON62926.2024.10647329.

[4] O. Boiko, A. Komin, R. Malekian, and P. Davidsson, "Edge-Cloud Architectures for Hybrid Energy Management Systems: A Comprehensive Review," *IEEE Sensors Journal*, vol. 24, no. 10, pp. 15748–15772, May 2024, https://doi.org/10.1109/JSEN.2024.3382390.

[5] F. Izourane, S. Ardchir, S. Ounacer, and M. Azzouazi, "Smart Campus Based on AI and IoT in the Era of Industry 5.0: Challenges and Opportunities," in *Industry 5.0 and Emerging Technologies*, vol. 565, A. Chakir, R. Bansal, and M. Azzouazi, Eds. Springer Nature Switzerland, 2024, pp. 39–57.

[6] B. Lounnas, M. Benazi, and M. Kamel, "A robust two-step algorithm for community detection based on node similarity," *The Journal of Supercomputing*, vol. 80, no. 16, pp. 23592–23608, Nov. 2024, https://doi.org/10.1007/s11227-024-06328-x.

[7] A. A. A. Gad-Elrab, A. S. Alsharkawy, M. E. Embabi, A. Sobhi, and F. A. Emara, "Adaptive Multi-Criteria-Based Load Balancing Technique for Resource Allocation in Fog-Cloud Environments," *International journal of Computer Networks & Communications*, vol. 16, no. 1, pp. 105–124, Jan. 2024, https://doi.org/10.5121/ijcnc.2024.16107.

[8] M. Fahimullah, S. Ahvar, M. Agarwal, and M. Trocan, "Machine learning-based solutions for resource management in fog computing," *Multimedia Tools and Applications*, vol. 83, no. 8, pp. 23019–23045, Aug. 2023, https://doi.org/10.1007/s11042-023-16399-2.

[9] Y. Zhao, W. Li, F. Liu, J. Wang, and A. M. Luvembe, "Integrating heterogeneous structures and community semantics for unsupervised community detection in heterogeneous networks," *Expert Systems with Applications*, vol. 238, Mar. 2024, Art. no. 121821, https://doi.org/10.1016/j.eswa.2023.121821.

[10] K. Tocze and S. Nadjm-Tehrani, "ORCH: Distributed Orchestration Framework using Mobile Edge Devices," in *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, Larnaca, Cyprus, May 2019, pp. 1–10, https://doi.org/10.1109/CFEC.2019.8733152.

[11] M. Khaddafi and F. Ferdiansyah, "Analisis Perbandingan Return Dan Risk (Studi Pada Saham Syariah Dan Saham Konvensional LQ45 Periode 2012-2016)," *Jurnal Akuntansi dan Keuangan*, vol. 5, no. 1, Feb. 2017, Art. no. 33, https://doi.org/10.29103/jak.v5i1.1811.

[12] P. Periasamy *et al.*, "ERAM-EE: Efficient resource allocation and management strategies with energy efficiency under fog–internet of things environments," *Connection Science*, vol. 36, no. 1, Dec. 2024, Art. no. 2350755, https://doi.org/10.1080/09540091.2024.2350755.

[13] W. Bai and Y. Wang, "Jointly Optimize Partial Computation Offloading and Resource Allocation in Cloud-Fog Cooperative Networks," *Electronics*, vol. 12, no. 15, Jan. 2023, Art. no. 3224, https://doi.org/10.3390/electronics12153224.

[14] V. Karagiannis and S. Schulte, "Distributed algorithms based on proximity for self-organizing fog computing systems," *Pervasive and Mobile Computing*, vol. 71, Feb. 2021, Art. no. 101316, https://doi.org/10.1016/j.pmcj.2020.101316.

[15] V. A. Traag, L. Waltman, and N. J. Van Eck, "From Louvain to Leiden: guaranteeing well-connected communities," *Scientific Reports*, vol. 9, no. 1, Mar. 2019, Art. no. 5233, https://doi.org/10.1038/s41598-019-41695-z.

[16] J. Liu, J. Wang, and B. Liu, "Community Detection of Multi-Layer Attributed Networks via Penalized Alternating Factorization," *Mathematics*, vol. 8, no. 2, Feb. 2020, Art. no. 239, https://doi.org/10.3390/math8020239.

[17] J. Vergara, J. Botero, and L. Fletscher, "A Comprehensive Survey on Resource Allocation Strategies in Fog/Cloud Environments," *Sensors*, vol. 23, no. 9, Jan. 2023, Art. no. 4413, https://doi.org/10.3390/s23094413.

[18] V. K. Quy, A. Chehri, N. M. Quy, V.-H. Nguyen, and N. T. Ban, "An Efficient Routing Algorithm for Self-Organizing Networks in 5G-Based Intelligent Transportation Systems," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1757–1765, Feb. 2024, https://doi.org/10.1109/TCE.2023.3329390.

[19] A. Góes-Neto *et al.*, "Comparison of complex networks and tree-based methods of phylogenetic analysis and proposal of a bootstrap method," *PeerJ*, vol. 6, Feb. 2018, Art. no. e4349, https://doi.org/10.7717/peerj.4349.

[20] I. A. Reshi and S. Sholla, "Securing IoT data: Fog computing, blockchain, and tailored privacy-enhancing technologies in action," *Peer-to-Peer Networking and Applications*, vol. 17, no. 6, pp. 3905–3933, Nov. 2024, https://doi.org/10.1007/s12083-024-01801-z.

[21] A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane, "Resource Provisioning in Edge Computing for Latency-Sensitive Applications," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11088–11099, Jul. 2021, https://doi.org/10.1109/JIOT.2021.3052082.