WFQ-Based SLA-Aware Edge Applications Provisioning

Pedro Henrique Sachete Garcia[1][a], Arthur Francisco Lorenzon[3][b], Marcelo Caggiani Luizelli[1][c], Paulo Silas Severo de Souza[1][d] and Fábio Diniz Rossi[2][e]

[1]*Federal University of Pampa, Brazil*

[2]*Federal Institute Farroupilha, Brazil*

[3]*Federal University of Rio Grande do Sul, Brazil*

fl

Keywords: Application Performance, Edge Provisioning, Resource Allocation.

Abstract: Provisioning delays can severely degrade the performance of real-time applications, especially in critical sectors such as healthcare, smart cities, and autonomous vehicles, where fast and reliable responses are essential. While managing data traffic effectively, existing flow scheduling techniques often fail to account for high-level metrics like Service-Level Agreements, leading to suboptimal prioritization of critical applications. This paper introduces a novel algorithm designed to optimize network bandwidth allocation for edge applications by incorporating SLA-based metrics. Evaluations demonstrate that our proposal outperforms conventional scheduling techniques. The results show that under varying infrastructure usage levels, our proposal consistently reduces provisioning times and minimizes delay violations for edge applications.

1 INTRODUCTION

The provisioning of edge applications is currently facing a pressing challenge (Temp et al., 2023). The increasing complexity and volume of connected devices are demanding fast and efficient responses. Any delay in this provisioning process can compromise the performance of applications that rely on low latency and high availability, which are essential characteristics in edge computing environments (Temp. et al., 2023). In scenarios with high demand for processing and data transfer, even a slight delay in provisioning can lead to service degradation, performance drops, interruptions, or even system failures. These outcomes are unacceptable in critical contexts such as in industries and smart cities. The delay in provisioning can be caused by infrastructure overload, lack of resource prioritization, or poor data flow management (Souza. et al., 2022).

Flow scheduling techniques are one approach to addressing this demand for prioritization in edge application provisioning (Cho and Easwaran, 2020). These techniques are designed to dynamically manage and allocate data flows, prioritizing those that require more bandwidth and minimal latency. In edge computing, flow scheduling can organize data traffic, allowing critical applications, such as the ones mentioned, to gain priority access to network resources (Tang et al., 2021). This way, it is possible to optimize provisioning time, minimizing delays for the applications that need the fastest response. However, most current flow scheduling techniques have a significant limitation: they do not consider high-level metrics, such as compliance with the service level agreements (SLAs) of the applications. These SLAs establish performance and availability parameters that must be maintained to ensure the quality of the service provided (Ramneek and Pack, 2021). Ignoring these metrics can result in poor prioritization of flows, where critical applications may not receive the proper attention in terms of resource allocation, even though they theoretically need wider bandwidth or lower latency.

The article introduces a tool called SLWQ (Service-Level Weighted Queuing), which focuses on efficient network bandwidth allocation to accelerate the transfer of container images over the network, thereby improving the speed and efficiency of provi-

[a] https://orcid.org/0009-0005-7487-854X
[b] https://orcid.org/0000-0002-2412-3027
[c] https://orcid.org/0000-0003-0537-3052
[d] https://orcid.org/0000-0003-4945-3329
[e] https://orcid.org/0000-0002-2450-1024

Garcia, P. H. S., Lorenzon, A. F., Luizelli, M. C., Severo de Souza, P. S. and Rossi, F. D.
WFQ-Based SLA-Aware Edge Applications Provisioning.
DOI: 10.5220/0013268000003950
In *Proceedings of the 15th International Conference on Cloud Computing and Services Science (CLOSER 2025)*, pages 159-166
ISBN: 978-989-758-747-4; ISSN: 2184-5042
Copyright © 2025 by Paper published under CC license (CC BY-NC-ND 4.0)

sioning edge applications. SLWQ is an adaptation of the Weighted Fair Queueing (WFQ) algorithm, incorporating the applications SLA as a weight to allocate network resources more effectively. By prioritizing applications with higher service requirements, SLWQ ensures that critical applications receive more bandwidth, leading to faster provisioning times. Although WFQ was originally developed for managing network flow control, SLWQ extends this concept to resource allocation, demonstrating its viability in contexts that require high-level metrics to manage resource distribution and guarantee compliance with service level agreements (SLAs).

This paper is organized as follows: Section 2 presents Edge conceptualization, applications provisioning ans SLA. Section 3 presents the problem definition. Section 4 presents SLWQ. Section 5 presents evaluations and results discussion. Section 6 presents the related work. Section 7 presents conclusions and future work.

2 BACKGROUND

Edge computing architecture is an infrastructure model aimed at decentralizing data processing and storage, bringing them closer to end devices and users (Satyanarayanan et al., 2009). Instead of relying on centralized servers in a remote data center, edge computing uses distributed nodes close to data sources to perform tasks and store information. This approach reduces latency, improves efficiency, and enables the execution of applications that require real-time response. Edge computing is particularly suitable for use cases involving the Internet of Things (IoT) (de Souza et al., 2017), where a large number of devices generate and consume data in nearby locations. Thus, edge architecture complements cloud computing by keeping processing power close to the user without the need to continuously transmit large volumes of data to distant servers (Pitstick et al., 2024).

Application provisioning in edge computing involves several essential steps and components to ensure that services are made available quickly and effectively. The container-based architecture plays a crucial role in this process (Hu et al., 2023). Containers are lightweight units that encapsulate an application and all its dependencies, ensuring it runs consistently in any environment (Xavier et al., 2013). By using containers in edge computing, it is possible to ensure that applications are distributed and deployed quickly, facilitating scalability and resource management. Container registries are repositories where container images are stored and managed. These registries can be accessed by edge nodes, which download the necessary images for application provisioning. Using local and distributed registries is an effective strategy to reduce provisioning latency since images are closer to the nodes that request them, avoiding the transfer time from a centralized server (Temp et al., 2024).

An aspect of successful application provisioning in edge computing is the efficient management of network channels and flows. Since the edge operates with limited resources, the available bandwidth may be scarce and needs to be used as efficiently as possible (Kassir et al., 2020). A well-managed network is essential to ensure that edge computing delivers on its promises of low latency and high efficiency, because network flows directly influence the ability to efficiently provision applications in edge computing, allowing them to be delivered as quickly as possible and within the parameters required by service level agreements (Karagiannis and Papageorgiou, 2017).

3 PROBLEM DEFINITION

Figure 1 illustrates three different flow control policies for computer networks, each dealing with four data flows competing for the same communication channel. The policies shown are Equal Share, Max-Min Fairness, and Weighted Fair Queueing, each with its own approach to allocating the available channel resources, resulting in different impacts on the performance and satisfaction of competing flows with varying demands.

In the Equal Share policy (Zhao and Chen, 2001), all flows receive an equal share of the total communication channel capacity, regardless of their individual demands. Thus, each flow receives 2.5 units of the total bandwidth of 10 units. While this approach is extremely simple and fair in terms of equality, it does not take into account the differences in each flow's needs. For instance, the flow with a demand of 2 receives 2.5 units, which exceeds its actual need, while other flows with higher demands may not be fully satisfied. This allocation may result in inefficient resource usage since a flow with low demand ends up receiving more than necessary, while more demanding flows remain underserved.

The Max-Min Fairness policy (Min et al., 2009), in turn, seeks to maximize the allocation for smaller flows, ensuring that each flow receives the maximum possible amount without compromising the minimum required for other flows. In the presented example, the flow with a demand of 2 receives exactly 2 units, fully

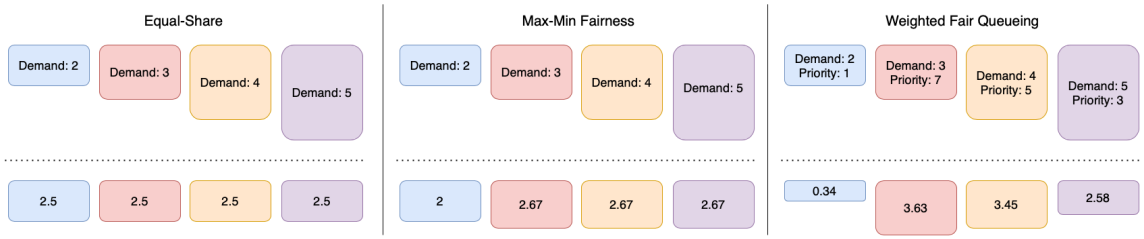


Figure 1: Comparison of flow scheduling policies.

meeting its need. The remaining flows receive 2.67 units each, as, after satisfying the demand of flow 2, the rest of the bandwidth is distributed equally among the remaining flows. While this approach is fairer by considering the specific needs of the flows, it still does not take into account differentiation of priorities among flows. Moreover, the performance of larger flows is impacted because the policy focuses on protecting smaller flows, even if this results in limiting flows with higher demands.

The third policy presented is Weighted Fair Queueing (Al-Sawaai et al., 2010), which uses different weights for each flow, reflecting their relative demand. In this example, flows are weighted according to their demands and priorities: the flow with a demand of 3 receives more resources (3.63 units), followed by the flow with a demand of 4 (3.45 units), then the flow with a demand of 5 (2.58 units), and lastly the flow with a demand of 2 (0.34 units). This approach tries to provide proportional treatment to flows based on their demands and priorities and proves more efficient from a bandwidth utilization perspective, as it considers the specific needs of each flow. However, like the previous policies, Weighted Fair Queueing ignores the importance of differentiating flows based on criteria such as the criticality of the applications or the context in which they are running. Thus, it may not be the best approach for applications that need faster provisioning or have specific quality of service requirements.

Although these policies are effective in traditional network environments, they are not suitable for edge computing environments, where it is necessary to consider factors beyond equitable or proportional resource distribution. In an edge computing scenario, it is essential to prioritize certain applications, such as critical ones that need to be provisioned quickly to ensure a good user experience or to meet specific operational requirements. Additionally, the size of the data being transferred, such as large images, must be taken into account, especially in channels with limited bandwidth, where an allocation based solely on demand may not meet the real needs of more sensitive applications.

4 SLWQ: SERVICE-LEVEL WEIGHTED QUEUING

This work proposes an alternative approach to network flow management, called ”Service-Level Weighted Queueing” (SLWQ), which expands upon the concept of the Weighted Fair Queueing (WFQ) technique. While WFQ uses weights proportional to the demands of the flows to determine bandwidth distribution, SLWQ incorporates weights derived from high-level metrics, such as application service level agreements (SLA). This approach allows allocations to be adjusted not only to the flows' demands but also to the specific requirements of each application, aligning resource allocation with the characteristics and needs of each flow.

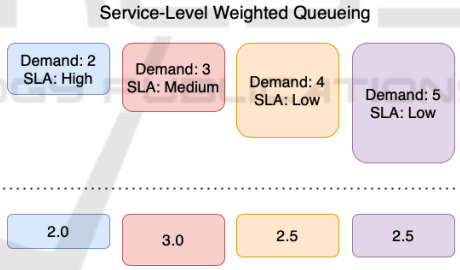


Figure 2: Service-Level Weighted Queuing policy.

In Figure 2, we can see four distinct application flows, each with a specific demand and SLA. The SLWQ policy considers both the application demand and SLA to determine resource allocation. For instance, the application with demand 2 and High SLA receives an allocation of 2.0 units (100% of the requested demand), while the application with demand 3 and Medium SLA receives 3.0 units (also 100% of the requested demand). The remainder is divided among applications with low SLA. This discrepancy occurs because SLA plays a crucial role in determining the weight assigned to each application, directly impacting how much bandwidth each one receives. Unlike traditional WFQ, which distributes resources based on proportional demands, SLWQ prioritizes application flows with higher levels of importance, as

indicated by the SLA metric, even if it means partially sacrificing the allocation of lower-SLA application flows.

In environments where multiple services compete for the same resources, it is essential for network management to recognize the difference in importance between these services. SLWQ allows more sensitive applications to be prioritized, while less critical applications can be delayed or receive fewer resources as long as this does not significantly compromise their operation. Also, SLWQ aims to optimize the use of network resources in situations where application demands and requirements are heterogeneous and often conflicting. In edge computing environments, where resources are limited and rapid response is essential, the use of SLWQ can ensure that critical applications receive the necessary treatment to meet their SLA requirements, while maximizing overall network efficiency and maintaining satisfactory performance for all applications involved.

5 EVALUATION AND DISCUSSION

In this section, we evaluate the performance of the SLWQ (Service-Level Weighted Queuing) algorithm compared to other network provisioning policies by analyzing the results of provisioning time (as seen in Figure 3) and delay violations (illustrated in Figure 4). Our evaluation focuses on three distinct levels of prioritization: low, medium, and high priority applications. As expected, SLWQ effectively prioritizes high-priority applications over medium and low-priority ones, demonstrating significant improvements over traditional queuing policies.

5.1 SLWQ

The Service-Level Weighted Queueing (SLWQ) presented in Algorithm 1 is an approach designed to manage bandwidth allocation efficiently, driven by service levels (SLA), and catering to the specific demands of each flow in a network. Table 1 consists of notations used in the Algorithm. The core of SLWQ lies in its ability to prioritize flows with higher criticality, as specified in the SLAs, while adjusting the allocation according to each flow's demand requirements. This process is achieved by combining weights based on SLA and relative demands, ensuring that the most critical flows receive greater priority and a proportionally more significant share of the available bandwidth.

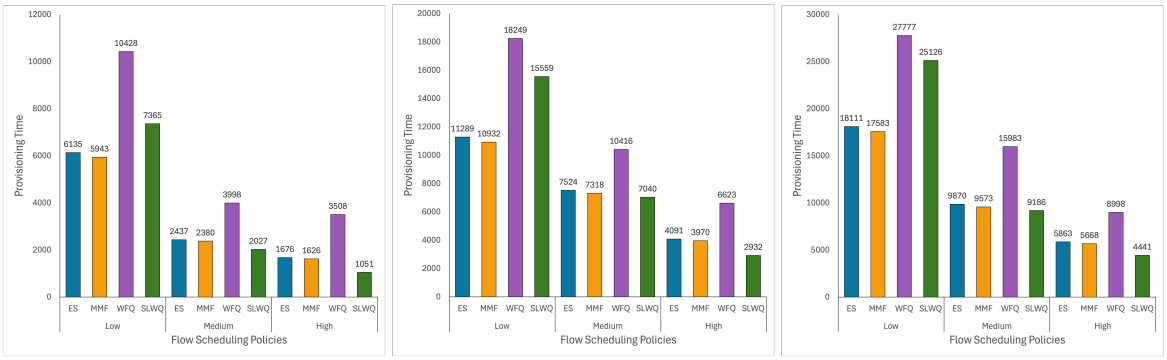
Table 1: SLWQ Notations.

Symbol	Description
cap	Total available bandwidth
dem	Array of demands for each flow
wgt	Array of priority weights for each flow
exp	Exponent applied to each demand
alloc	Array of allocated bandwidths
valD	List of valid demands
totW	Total weight for allocation
capRem	Remaining capacity

The algorithm's input consists of several essential parameters. First, it takes the total available bandwidth, which represents the maximum resource that can be distributed among flows. Next, it uses an array of demands, where each position represents the specific bandwidth demand of a flow. There is also an array of weights, where each weight is associated with the priority of the corresponding flow, indicating the relative importance of the flow based on its SLA. Additionally, the algorithm uses a parameter called "exp," an exponent applied to each demand to create a weighted distribution; this configuration allows for adjusting the allocation of resources according to each demand and priority profile.

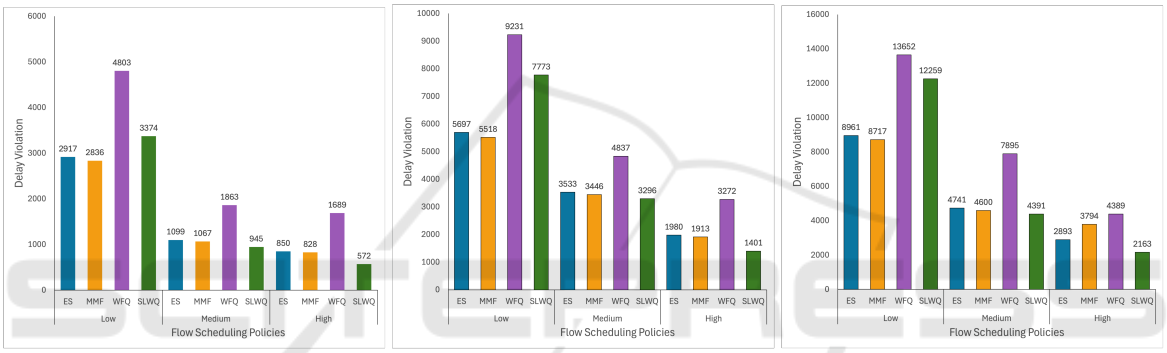
Algorithm 1: Service-Level Weighted Queueing (SLWQ).

Input:
  $cap \rightarrow$ Total available bandwidth
  $dem[] \rightarrow$ Array of demands for each flow
  $wgt[] \rightarrow$ Array of priority weights for each flow
  $exp \rightarrow$ Exponent for weighted allocation
Output:
  $alloc[] \rightarrow$ Array of allocated bandwidths for each flow
Initialize *alloc* as a zero array of the same length as *dem*
Define *valD* as the list of tuples (d, w, i) where each $d > 0$, with w as the corresponding weight and i as the index
Set $totW \leftarrow \sum_{(d,w,i) \in valD} (d^{\exp}) \times w$
Initialize $capRem \leftarrow cap$
for each (d, w, i) in *valD* **do**
  Compute $alloc \leftarrow \min\left(\frac{capRem \times (d^{\exp} \times w)}{totW}, d\right)$
  Update $alloc[i] \leftarrow alloc[i] + alloc$
  Update $capRem \leftarrow capRem - alloc$
  Update $totW \leftarrow totW - (d^{\exp} \times w)$
  if $totW \leq 0$ **or** $capRem < 0$ **then**
    break
  end if
end for
return *alloc*



(a) Infrastructure usage: 25%. (b) Infrastructure usage: 50%. (c) Infrastructure usage: 75%.

Figure 3: Provisioning Time: Number of violations of the SLA established as acceptable for the migration or provisioning of each application. When this threshold is crossed, it means that users have moved to a location less close to where the service is provisioned, and it means that a new service must be provisioned on another edge node that is closer to such users.



(a) Infrastructure usage: 25%. (b) Infrastructure usage: 50%. (c) Infrastructure usage: 75%.

SCIENCE AND TECHNOLOGY PUBLICATIONS

Figure 4: SLA Delay Violations: Number of violations of the acceptable application delay threshold. When this threshold is crossed, it means that the service is too far from the registry located in the core of the cloud.

The SLWQ algorithm begins by initializing an empty bandwidth allocation array and creating a list of valid demands, which includes each flow's demand, weight, and index, filtering out flows with no demand. It calculates the total weight as the sum of each demand raised to an exponent multiplied by its weight. The remaining bandwidth capacity is initialized to the total capacity. In the main loop, the algorithm iterates through each flow, calculating the bandwidth allocation as a proportion of the remaining capacity based on the flow's relative weight and demand. To prevent over-allocation, the allocated bandwidth is capped at the flow's demand. The allocated bandwidth is stored in the array, and the remaining capacity and total weight are updated accordingly. The loop ends when either all demands are met or the bandwidth is fully allocated. The algorithm then returns the final array of allocated bandwidth values, ensuring prioritized distribution based on demands and SLAs, while respecting total bandwidth limits and individual flow demands.

5.2 Provisioning Time Evaluation

Figure 3 presents the provisioning time across three levels of infrastructure usage: 25%, 50%, and 75%. Across all tests, SLWQ consistently provides the fastest provisioning time for high-priority applications, outperforming Equal Share, Max-Min Fairness, and even Weighted Fair Queuing (WFQ). This superior performance can be attributed to SLWQ's ability to dynamically adjust resource allocation based on both the priority and the demand of applications.

At 25% infrastructure usage, all policies perform relatively well due to the availability of ample resources. However, even under these low-stress conditions, SLWQ demonstrates its advantage by allocating a higher proportion of bandwidth to high-priority applications, ensuring their provisioning times remain minimized. The differences between SLWQ and other

Table 2: Comparison between SLWQ and Evaluated Scheduling Policies.

Feature/Criteria	SLWQ	WFQ	Max-Min Fairness	Equal Share
Priority-based Resource Allocation	Yes	Partial	No	No
Latency Minimization	High	Medium	Low	Low
Bandwidth Utilization Efficiency	High	Medium	Low	Low
Handling of Large Data Volumes	Efficient	Moderate	Poor	Poor
Real-time Application Support	Strong	Moderate	Weak	Weak
Adaptation to Infrastructure Load	Yes	No	No	No
SLA Support	Yes	No	No	No

policies become more pronounced as infrastructure usage increases to 50% and 75%. Under higher loads, the competing policies—particularly Equal Share and Max-Min Fairness—fail to adequately prioritize high-priority applications, resulting in longer provisioning times for these critical services. In contrast, SLWQ maintains a strategic allocation of resources, ensuring that the most important applications continue to receive the necessary bandwidth, thus minimizing delays.

This advantage becomes even more critical under the 75% infrastructure usage scenario, where resource contention is at its peak. SLWQ's prioritization mechanism ensures that the high-priority applications, which are often latency-sensitive, experience the smallest provisioning delays. In comparison, WFQ, while better than Equal Share and Max-Min Fairness, still cannot adapt as effectively to high-priority demands since it only accounts for demand and not priority, which is the key differentiator in SLWQ's design.

5.3 Delay Violations Evaluation

Figure 4 evaluates delay violations across the same levels of infrastructure usage. Delay violations occur when an application is not provisioned within the expected time frame, potentially leading to performance degradation. Similar to the provisioning time results, SLWQ exhibits a clear advantage, particularly in reducing delay violations for high-priority applications.

At 25% infrastructure usage, delay violations are minimal for all policies, as resources are abundant. However, as usage climbs to 50% and 75%, SLWQ maintains a lower incidence of delay violations compared to the other policies. For high-priority applications, SLWQ nearly eliminates delay violations, a critical achievement for real-time or mission-critical services, such as healthcare monitoring or autonomous vehicle data processing.

The failure of Equal Share and Max-Min Fairness to appropriately prioritize high-priority traffic results in significant delay violations, particularly under heavy infrastructure load. These policies, which either distribute resources equally or favor smaller demands, cannot distinguish the urgency of high-priority applications, leading to unacceptable delays. Although WFQ performs better by allocating more bandwidth to larger demands, it still results in a substantial number of delay violations under high load, as it lacks the application-SLA focus that SLWQ implements.

5.4 Discussion

The key strength of SLWQ lies in its ability to balance the competing demands of multiple applications while ensuring that the highest-priority ones receive preferential treatment. In contrast to Equal Share, which treats all applications equally regardless of their importance, and Max-Min Fairness, which prioritizes smaller flows, SLWQ intelligently adjusts its resource allocation to reflect both the application's demand and its criticality. This results in significantly better provisioning times and fewer delay violations for high-priority applications, especially in scenarios with limited resources.

WFQ, while closer to SLWQ in terms of accounting for the differing demands of applications, still falls short in environments where application priority must be considered. WFQ allocates bandwidth proportionally based on demand but does not factor in the criticality of each application. This limitation is particularly evident under high infrastructure usage, where SLWQ's additional consideration of service-level agreements (SLAs) allows it to outperform WFQ.

SLWQ's ability to incorporate high-level metrics such as application priority into the queuing mechanism ensures that the most important services—those with stringent latency and performance requirements—are given priority over less critical services. This approach maximizes overall network efficiency and service quality, particularly in edge computing environments where fast provisioning is often essential for maintaining the functionality of real-time applications.

The results demonstrate that SLWQ provides a robust and effective solution for managing network provisioning in edge computing environments. Its ability to adapt to varying levels of infrastructure usage while maintaining a focus on application priority makes it particularly suited to high-demand, resource-constrained environments. As edge computing con-

Table 3: Comparison between Related Work and SLWQ.

Feature/Work	Shinohara et al. (1997)	Jiang et al. (2000)	Jutila (2016)	Chen et al. (2021)	SLWQ (2024)
Type of System	Edge modules	Cellular Networks	IoT Edge Router	Cooperative Games	Edge Applications
Algorithm	WFQ	WFQ	Fuzzy Weighted Queuing	Nucleolus Computation	SLA-based WFQ
QoS Guarantees	Yes	Yes	Yes	No	Yes
Service Classes	Multiple traffic classes	Multiple service classes	Dynamic adaptation	Coalition fairness	High-level SLA priorities
Priority-based Allocation	Partial	Partial	Yes	No	Yes
Context Awareness	No	No	Yes	Yes (Cooperative)	Yes (SLA-driven)
Handling of Bursty Traffic	No	Yes	Yes	No	Yes
Dynamic Bandwidth Adjustment	No	No	Yes	No	Yes

tinues to evolve, the need for intelligent resource allocation mechanisms like SLWQ will become increasingly important. Table 1 summarizes the differences between SLWQ and other flow schedulers, in several performance evaluation metrics.

6 RELATED WORK

Weighted fair queueing (WFQ) is a scheduling algorithm widely studied in network traffic management. (Shinohara et al., 1997) proposed a growable switch architecture with multiple modules, including Edge modules with large buffers that utilize a weighted fair queueing scheduler for open-loop traffic. This approach allows for allocating equivalent bandwidth based on call admission control, ensuring quality of service (QoS) guarantees for different traffic classes. (Jiang et al., 2000) evaluated the performance of the weighted fair queueing algorithm in providing multiple service classes for bursty data traffic in cellular networks, similar to an Edge system. The study focused on packet scheduling to achieve fair bandwidth sharing among different types of traffic, highlighting the importance of efficient queueing mechanisms in wireless communication systems.

Furthermore, (Jutila, 2016) discussed the importance of edge computing in managing network resources, traffic prioritizations, and security in the Internet of Things (IoT) context. The author introduced adaptive edge computing solutions, including fuzzy weighted queueing (FWQ), to monitor and react to changes in the quality of service within heterogeneous networks, emphasizing the role of edge computing in enhancing network performance. In Cooperative games, (Chen et al., 2021) explored the computation of the nucleolus in various game scenarios, including fractional edge cover games on weighted graphs. The study focused on finding fair divisions of payoffs among coalition members, highlighting the significance of fair resource allocation in cooperative settings.

Table 3 summarizes all related works and compares them with SLWQ, comparing some metrics. Altough the literature review showcases the significance of weighted fair queueing in optimizing network traffic management, ensuring QoS guarantees, and facilitating fair resource allocation in diverse communication systems, none of the literature considers high-level metrics such as those service-level agreements aimed at ensuring quality of service.

7 CONCLUSION AND FUTURE WORK

The problem addressed in this work revolves around the inadequacies of traditional flow scheduling techniques in edge computing environments. These techniques, including Equal Share, Max-Min Fairness, and Weighted Fair Queueing (WFQ), either fail to account for the specific demands of applications or do not prioritize them according to their criticality and Service-Level Agreements (SLAs). In edge computing environments where resources are constrained and multiple applications are competing for bandwidth, these approaches often result in inefficiencies such as increased provisioning times and higher rates of delay violations, particularly for high-priority applications.

In this work, we proposed the Service-Level Weighted Queuing (SLWQ) algorithm, a novel approach designed to optimize network bandwidth allocation for edge computing. The results of our evaluations highlight the advantages of SLWQ over traditional policies. In our tests, which involved three distinct levels of application SLA (low, medium, and high), SLWQ consistently outperformed other policies, especially under high infrastructure usage. When provisioning time was measured (as shown in Figure 3), SLWQ demonstrated a significant reduction in the time required to provision high-SLA applications, particularly when compared to Equal Share, Max-Min Fairness, and WFQ. Additionally, SLWQ proved to be highly effective in minimizing delay violations, as depicted in Figure 4. Delay violations occur when an application fails to be provisioned within the expected time, which can lead to serious consequences in time-sensitive applications.

Future work could explore the integration of additional factors, such as energy efficiency or predictive analytics, into SLWQ's prioritization framework.

ACKNOWLEDGEMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

REFERENCES

Al-Sawaai, A., Awan, I. U., and Fretwell, R. (2010). Performance of weighted fair queuing system with multiclass jobs. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 50–57.

Chen, N., Lu, P., and Zhang, H. (2021). Computing the nucleolus of matching, cover and clique games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1319–1325.

Cho, H. and Easwaran, A. (2020). Flow network models for online scheduling real-time tasks on multiprocessors. *IEEE Access*, 8:172136–172151.

de Souza, P. S. S., dos Santos Marques, W., Rossi, F. D., da Cunha Rodrigues, G., and Calheiros, R. N. (2017). Performance and accuracy trade-off analysis of techniques for anomaly detection in iot sensors. In *2017 International Conference on Information Networking (ICOIN)*, pages 486–491.

Hu, S., Shi, W., and Li, G. (2023). Cec: A containerized edge computing framework for dynamic resource provisioning. *IEEE Transactions on Mobile Computing*, 22(7):3840–3854.

Jiang, Z., Chang, L. F., and Shankaranarayanan, N. (2000). Providing multiple service classes for bursty data traffic in cellular networks. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 3, pages 1087–1096 vol.3.

Jutila, M. (2016). An adaptive edge router enabling internet of things. *IEEE Internet of Things Journal*, 3(6):1061–1069.

Karagiannis, V. and Papageorgiou, A. (2017). Network-integrated edge computing orchestrator for application placement. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–5.

Kassir, S., Veciana, G. d., Wang, N., Wang, X., and Palacharla, P. (2020). Service placement for real-time applications: Rate-adaptation and load-balancing at the network edge. In *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 207–215.

Min, Z., chunming, W., Ming, J., and Jing, Y. (2009). A resource management algorithm based on multi-path max-min fairness. In *2009 First International Conference on Future Information Networks*, pages 76–80.

Pitstick, K., Novakouski, M., Lewis, G. A., and Ozkaya, I. (2024). Defining a reference architecture for edge systems in highly-uncertain environments. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*, pages 356–361.

Ramneek and Pack, S. (2021). A credible service level agreement enforcement framework for 5g edge. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.

Shinohara, M., Fan, R., Mark, B., Ramamurthy, G., Suzuki, H., and Yamada, K. (1997). Multiclass large scale atm switch with qos guarantee. In *Proceedings of ICC'97 - International Conference on Communications*, volume 1, pages 453–461 vol.1.

Souza., P., Ângelo Crestani., Rubin., F., Ferreto., T., and Rossi., F. (2022). Latency-aware privacy-preserving service migration in federated edges. In *Proceedings of the 12th International Conference on Cloud Computing and Services Science - CLOSER*, pages 288–295. INSTICC, SciTePress.

Tang, H., Wang, Y., and Chen, X. (2021). Network-flow algorithms for virtual service placements in mobile edge networks. In *2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pages 364–368.

Temp., D., Capeletti., I., Goes de Castro., A., Silas Severo de Souza., P., Lorenzon., A., Luizelli., M., and Rossi., F. (2023). Latency-aware cost-efficient provisioning of composite applications in multi-provider clouds. In *Proceedings of the 13th International Conference on Cloud Computing and Services Science - CLOSER*, pages 297–305. INSTICC, SciTePress.

Temp, D. C., da Costa, A. A. F., Vieira, A. N. C., Oribes, E. S., Lopes, I. M., de Souza, P. S. S., Luizelli, M. C., Lorenzon, A. F., and Rossi, F. D. (2024). Maper: mobility-aware energy-efficient container registry migrations for edge computing infrastructures. *The Journal of Supercomputing*, 81(1):15.

Temp, D. C., de Souza, P. S. S., Lorenzon, A. F., Luizelli, M. C., and Rossi, F. D. (2023). Mobility-aware registry migration for containerized applications on edge computing infrastructures. *Journal of Network and Computer Applications*, 217:103676.

Xavier, M. G., Neves, M. V., Rossi, F. D., Ferreto, T. C., Lange, T., and De Rose, C. A. F. (2013). Performance evaluation of container-based virtualization for high performance computing environments. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 233–240.

Zhao, Y. and Chen, C. (2001). The algorithm and model of trash: A scheme enforcing equal share. In *Proceedings 2001 International Conference on Computer Networks and Mobile Computing*, pages 425–432.