Towards efficient service provisioning in Edge Computing

*
*
*
*
*

***Abstract*—The growing demand for digital services has made the use of Edge Computing increasingly necessary. However, for its efficient usage, mechanisms that take into account the distributed nature of its components are needed. Thus, this paper indicates 3 challenges for the efficient use of Edge Computing. The first is the forwarding of the user to the nearest server. The second is the instantiation of services considering the volume and origin of demand, and the resources needed to meet it. The third is that the request chaining should be done considering the infrastructure topology. To address the first challenge, it implements Topology Aware DNS. A proposal to deal with the second challenge, and main contribution of the research, is presented in sequence. For this, it formulates 5 indicators that synthesize all the necessary information to describe the state of the infrastructure and the services provided by it. Then, it indicates how these indicators can be used by a Machine Learning algorithm to perform the orchestration of the allocation of instances of these services. Additionally, it also indicates possible strategies to address the third challenge. The analysis of the experiments described in this paper focuses on evaluating the benefit of the approach that addresses the first challenge. The results obtained show that 76.99% of the requests were optimally or well answered, and only 1.51% were classified as bad. Future work will address the evaluation of the proposal that is targeted to address the second challenge.**

***Index Terms*—efficiency in edge computing, machine learning, resource allocation**

I. Introduction

Ubiquitous computing is a reality today. Through the use of smartphones it is possible to view multimedia content, perform financial transactions, and access numerous public services. The processing power that is built into these devices enables the capture and display of high quality video. Additionally, the included communication networks provide high bandwidth Internet access. In parallel, the use of IoT has been widely used in tracking the delivery of products in online commerce, as well as in agriculture, industry and the environment monitoring. Moreover, a resource that has been explored in the implementation of smart cities is the processing of images from monitoring cameras to improve the response time of services to attend incidents in urban centers.

Given this scenario, there is an increasing demand for more dynamic and interactive services, accentuating the need for a computing infrastructure adapted to this new context. Thus, infrastructures with elements closer to the network edge have been developed, known as Edge Computing. This strategy reduces latency in access, improving the responsiveness of these services. The use of this type of infrastructure is usually done in a complementary way to Cloud Computing, allowing to improve the logistics of service deployment in the infrastructure in order to maximize the Quality of Experience to the user.

Hence, the need for research on the optimization of service orchestration in distributed infrastructures is identified. Although similar issues have already been explored in Cloud Computing, it is distinguished by the presence of a less centralized infrastructure, associated with the innovations provided by container orchestration platforms (Kubernetes) and their integration with SDNs.

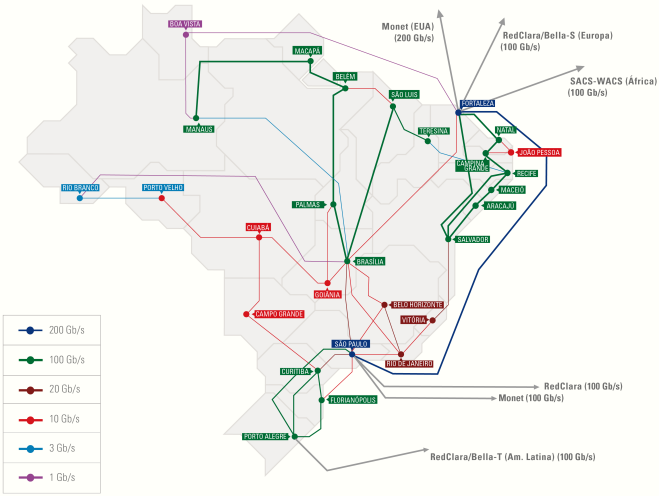

Fig. 1. RNP's network infrastructure (IPE network)

In order to contribute to this research, the present work is based on the use case of the video publication and distribution service developed to serve the academic community in Brazil, the eduplay[1]. This service currently has content from more than 350 universities and research institutions in Brazil. To meet this demand, it has a network that covers the entire Brazilian territory (figure 1). Over this infrastructure, it has

thanks to *

[1]http://eduplay.rnp.br

deployed a mesh of edge servers through which it distributes the content of this video platform, with a monthly rate of approximately 50 million requests, totaling more than 12 terabytes of delivered content. This service is a result of the research published in [1] [2] [3].

Therefore, this paper aims to present the first results obtained from the ongoing research that aims to maximize the efficiency of service deployment using Edge Computing. The contributions of this work are indicated below, as well as the structure of the paper.

Initially, we mapped out 3 challenges for the efficient delivery of services in an Edge Computing infrastructure (section II), and found solutions for them. The first one, which is Topology Aware DNS Description, can use a DNS-based solution (with ECS) [4], which is combined with HTTP Redirect (III). The presentation of a novel proposal to address the second challenge based on synthesizing the state of the environment into 5 indicators to parameterize the orchestration of service instantiation through ML (section IV). The mapping of possible strategies to address the third challenge (section V). Analysis of the results obtained in an experiment to evaluate the use of Topology Aware DNS in addressing the first challenge (section VI). Finally, the VIII section presents the conclusions and future work.

II. Challenges for Efficiency in Edge Computing

In general, the challenges to efficiently deploy a service in an Edge Computing infrastructure are associated with three fundamental aspects:

The first challenge **(C1)** is about ensuring that the client connects (establishes the TCP connection) to the element closest to it, resulting in the lowest latency of the communication. Because, besides giving a better perception of responsiveness, it allows maximizing the throughput between these. In addition, it is important to ensure that it has the availability to meet the requests. Thus, it is necessary that these mechanisms can act in an integrated way with the orchestration platform so that the necessary precision and dynamism can be obtained.

The second challenge **(C2)** represents the ability to manage the allocation of the instances of the components that provide the services, considering: the volume, the origin of the demand and which resources/contents/services are accessed. The differential in the context of Edge Computing is that the greater distribution of elements reflects directly on the spatial variation of demand, since the origin of requests can alternate between micro-regions, which can be mapped as a single region in Cloud Computing contexts. So it' s necessary to have a mechanism capable of identifying these behaviors and mapping them into allocation models that allow finding the trade-off between QoE and resource usage optimization.

The last challenge **(C3)** aims to orchestrate the communication between the various elements used to serve the request, aiming to optimize the communication between them. Example, in the video distribution scenario, a request is initially served by a tcp/http proxy service. Then, it makes a request to the cache service; this one, in turn, if it does not have the content in its cache, requests to another cache server or directly to the source. Hence, it is not enough that the client request routing is optimized, the communication between all the elements that compose the service is optimized, considering the mesh of components distributed throughout the Edge Computing infrastructure.

III. Combining DNS with HTTP to meet challenge 1

One of the fundamental steps to address challenge 1 is the choice of the mechanism that will forward the client's request to the server designated to serve it. In general, three techniques can be used: Anycast, DNS, or HTTP Redirect.

The most efficient techniques in relation to network overhead is Anycast. However, its implementation depends on how the infrastructure is built, and the establishment of traffic exchange agreements with adjacent networks.

The DNS approach, meanwhile, allows for scenarios in which the use of Anycast is not feasible. That is, in cases in which it is not possible to apply these optimizations in the routing process.

HTTP Redirect, on the other hand, allows for more precise client identification. Nevertheless, it represents an overhead in the initialization of the communication (compared to Anycast or DNS). This approach is useful in cases where the content to be transmitted is high volume (such as video).

In [5] it is discussed how this challenge is addressed by the Hyper-Giants of the Internet (Akamai, Amazon, Facebook, Google, Microsfot and Netflix), indicating proposals and resulting impacts.

Thus, the choice of strategy depends on several factors, including the feasibility of implementation, associated with the mechanisms supported by the infrastructure provider.

A. DNS-ECS for small, HTTP Redirect for big

Given the context in which this research is focused, the approach based on combining DNS with HTTP Redirect was chosen.

DNS is used to provide access to services characterized by multiple contents, but that each content has a low volume of data, such as browsing a website. To improve the accuracy in discovering the user's location in this case, Topology Aware DNS was developed, described in the following section (III-B).

HTTP Redirect is used to handle requests associated with a large volume of data, such as in the case of video streaming access, as described in [3]. The choice of this approach aims to maximize the quality of user experience given that latency in communication has a significant impact on the transfer rate of this content.

B. Topology Aware DNS Operation

Topology Aware DNS is composed of three sub-modules: Topology Discovery, Service Discovery and Lookup Engine.

Topology Discovery is responsible for: (i) creating a graph of the topology using a mapping of the links that make up the infrastructure and (ii) a binary search structure of IP addresses

using a geolocation database of IP address blocks as reference. Service Discovery queries the monitoring system to identify which nodes have registered services and their availability level.

The Lookup Engine in turn is used in DNS requests to select the node and the corresponding IP address that will be used as a response. To do this it checks if the request has the EDNS Client Subnet information to use as a client address reference. If not, it uses the IP address of the DNS Resolver that forwarded the request. Based on this address, it defines the location of the user using the Topology Discovery search engine. Next, it queries Service Discovery to obtain the list of nodes that have the desired service and that have acceptable availability to serve the requests. Based on this information and the topology graph, it selects the node closest to the user's location using the Dijkstra algorithm. Finally, it returns the node's IP address information as a response.

C. Intra Node Load Balance

In a complementary way, it is important to use load balancing mechanisms within the node. The traditional way to implement this function is through network equipment, using VIP mechanisms. However, recent research has proposed to optimize this issue through the use of SDN, called fullstack-SDN [6]. The idea is to use SDN packet forwarding mechanisms in an integrated way with service orchestration functionalities to direct packets directly to the desired Pod via flow mapping. To implement these mechanisms, CNIs can be used to act in an integrated way with SDN controllers.

However, as in the context of the use case, the nodes that make up Edge Computing are distributed virtual machines using K3S. Thus, this aspect was implemented through the use of the ingress service.

IV. Synthesized indicators to deal challenge 2

The main research contribution associated with this paper is proposed to address challenge 2. To this end, it uses ML to perform the orchestration of the instances. The differential is in the proposition of 5 indicators that synthesize all relevant aspects for parameterization of the ML algorithm. The goal is to avoid the problem known as "curse of dimensionality" [7].

For this, the concepts of Measurement, Metrics and Indicators for software engineering are used [8]. Briefly, a Measure represents a quantitative indication of the extent, quantity, capacity, or size of some attribute. A Metric relates a set of Measures. The Indicator in turn is the combination of metrics that allow the behavior/performance of the observed element to be analyzed. That is, it can be the correlation between several instants of the same metric, showing its behavior, or the combination between different metrics, to identify the influence that the measured attributes have on each other.

Furthermore, some collected data can be used to represent the context of the monitored attributes, in order to be used for the classification of measurements. For example, the mapping of IP address blocks can be used to classify a request to a service region. This information can be obtained through a BGP advertisement agent or by processing geolocation tables. Such information is called Measurement Metadata and is used as labels by monitoring mechanisms.

A. Measures and Metadata

The Measures to be collected are essentially information extracted from the data collected from the infrastructure or the services that are instantiated in it. This data are the result of state monitoring, which are already expressed in Measures, or the logs from which Measures are extracted either directly (e.g. a value contained in the log) or by counting the number of logs with a certain characteristic, such as successfully answered requests.

B. Metrics

After the measures are collected, the data processing step is started for the production of Metrics. For this, formulas are used to correlate the previously extracted measures.

1) Proportion of use of computing resources: Represents how much of the total available of a resource (CPU, RAM, HD, Net, Load1 and Link) is being used, expressed as a percentage.

2) Transfer rate: Quantifies the amount of data transmitted over a period of time. Example, the rate of bytes transferred (and received) in one second is represented by TX (and RX).

3) Requests rate: Summarizes the amount of requests received by the application in a period of time (in minutes).

4) RTT and SRT Percentile: This Metric uses the concept of percentile to determine the reference value for analyzing the Network Latency (RTT) and Server Response Time response time (SRT). The idea is to identify the most appropriate value to represent samples in optimal conditions as well as to represent a deteriorated condition.

C. Indicators

The last stage of data preparation is the production of Indicators. The benefit in the use of indicators is that, regardless of which algorithm is used for decision making, the reduction in the number of variables to be used to calculate the decision represents a reduction in the cost of this operation, since it minimizes the complexity of ML operations.

Another important aspect is that it allows the association of the values of the Measures and Metrics that represent the attributes in their natural units of magnitude into appropriate quantifiers to be used by the ML algorithms. Example, a Reinforcement Learning algorithm needs to receive a value that represents the reward for the action taken. But for that, it needs a formula that associates the throughput of sending data to clients, the use of computational resources and the number of problem requests. In addition, the calculation of the indicator needs to be adjusted to suit the semantic aspects of the service, including its operational cost, through financial variables. In order to obtain results more adherent to the expected goals.

1) Availability: Reflects the condition of the computational resource that represents the major constraint (among resources) of a given element at a specific instant. This can be used both to represent the state of the server on which the applications that implement the services are instantiated, and also the element that represents the instance (e.g. container/pod) of the application that composes the service.

The objective of this metric is to allow the algorithm, instead of having to deal with all the Resource Usage Measures, to evaluate only the one that represents the biggest limitation at that moment.

To calculate this metric the Resource Usage Ratio Metrics are used through the following expression:

$$\mathbf{A}vailability = 100 - max\{CPU, RAM, HD, Net, Load\} \quad (1)$$

2) Demand: Indicator computed based on the rate of data transmitted exclusively to clients. This means that values resulting from demand generated by internal mechanisms, such as cache propagation between nodes, should be disregarded. Furthermore, a classification must be performed according to the coverage region to which the client that made the request belongs. In this way, it allows the identification of the origin of the demand and helps in the definition of which location needs the instantiation of a service element.

In [9] studies that exploit the rate of requests to measure demand are indicated. However, since the use case is associated with videos, the network is the computational resource that is most in demand. Thus, the attribute used to measure this aspect is data volume. Thus, the following equation is used to quantify the demand indicator.

$$\mathbf{D}emand_{region} = \sum_{n} TX_n | n \in region \quad (2)$$

3) Cost: Cost summarizes the operating cost of an instance of a given service into a single measurable value. Measures used are also cpu, memory, storage and network usage. However, the composition of this indicator is not about the available resources, but rather the nominal value of usage, e.g. in bytes of memory used.

Also, to aggregate the value of each resource fairly, a weight parameter is used for each type of resource. Because even though they are measured in similar units of magnitude, they represent different operating costs. This parameterization should be estimated for each usage scenario.

Thus, the argument M indicates the Measure that quantifies the use of the resource and W indicates the weight associated with the operational cost of allocating this resource.

$$\mathbf{C}ost = M_{cpu} \cdot W_{cpu} + M_{ram} \cdot W_{ram} + M_{hd} \cdot W_{hd} + M_{net} \cdot W_{net} + M_{load} \cdot W_{load} \quad (3)$$

4) Utility: The utility indicator allows to measure the benefit of the instantiation of an element at a given location. The computation of this indicator is different for each service, because the impact of each one's performance is different in the system.

In the case of the caching service, the main factor that determines the utility is the amount of requests that are met through the use of content already stored locally. That is, the difference between the rate of transmitted data and the rate of received data. This approach is more suitable than using cache hit because the cost of operation is related to the volume of data. Thus, it was identified that the formulation of this Identifier is best represented by the following equation.

$$\mathbf{U}tility_{cache} = (TX_{cache} - RX_{cache}).W_{net} \quad (4)$$

The usefulness of the proxy service can be measured by the reduction of network latency to the client. Since it is a factor that directly affects the transmission rate that the client reaches due to the TCP sliding window effect. For this, the latency metric (RTT) is used considering the P50th percentile, in conjunction with the rate of client requests, since it directly reflects the use of this service, hence its utility.

The associated weight to calculate the utility can be based on the impact of the SLA (Service Level Agreement) or the performance expectation, represented by W_{RTT}. Essentially, it serves as a weight to calculate the trade-off between the cost of instantiation of the service and its benefit at a given location.

$$\mathbf{U}tility_{proxy} = \frac{1}{RTT^{p50}}.req_{ok}.W_{RTT} \quad (5)$$

D. Perturbation

The disturbance indicator seeks to measure the impact of problematic conditions on a given element, which can be used for either an instance of the TCP/HTTP proxy service or the cache service. The value is represented by the sum between: (i) the rate between the server response time (SRT) of the P90th percentile and the P50th percentile and (ii) the rate between the requests handled with error and those handled successfully.

$$\mathbf{P}erturbation = \frac{SRT_{p90}}{SRT_{p50}} + \frac{(req_{err} + 1)}{req_{ok}} \quad (6)$$

This indicator is used as a penalty factor in the reinforcement function by the ML algorithm.

E. Machine Learning Modeling

Using these indicators to take action in orchestration is through the use of a Deep Reinforcement Learning (DRL) algorithm. This strategy has been indicated in recent studies [10] [11] [12]. Furthermore, a multi-agent approach is used. That is, each agent is responsible for making decisions in a restricted scope of action. This way, it is possible to reduce the amount of information needed to represent the state of the environment in which the agent will act and amount of possible actions. This strategy aims to align with the concept defined in [13] "Ignorance is a blessing" when exploring multi-agent approach. So, each agent, has only the environment information of its own node and some parameters that represent the neighboring (directly connected) nodes, according to the following representation.

$$\begin{aligned}
\text{States} &= \begin{cases}
\text{Node } \mathbf{A}\text{vailability} \\
\mathbf{A}\text{vailability of each service instance on the node} \\
\mathbf{D}\text{emand from the service regions bound to the node} \\
\text{Number of service instances in the node} \\
\text{Neighborhood map (directly connected nodes), relating:} \\
\text{- } \mathbf{A}\text{vailability of the neighbor node} \\
\text{- Proportion of use of the interconnection link} \\
\text{- } \mathbf{D}\text{emand of the neighboring service region} \\
\text{- Number of instances of the service in the neighboring node}
\end{cases} \\
\text{Actions} &= \begin{cases}
\text{Add a new service instance in the node} \\
\text{Remove a service instance in the node} \\
\text{Make no changes}
\end{cases} \\
\text{Reinforcement} &= \sum_{i}^{service} \left(\frac{U_i}{C.(1 + P_i)} \right)
\end{aligned} \quad (7)$$

V. Strategies to address challenge 3

The strategy in operation in the use case of this paper applies the model described in [3]. In short, it uses the strategy of recursive http requests between the nodes of the content distribution network. The routing path is defined by an element called Maestro. This generates a URL in which the necessary information is encoded so that the elements that compose the service can reach the source of the content.

As part of the evolution of this strategy, it was proposed in [14] mechanisms for integration with SDNs. Thus, it presents an architecture that creates an abstraction layer between the CDN and the network infrastructure through an extension to the ALTO protocol [15], the ALTO-Intents. The goal is to allow, in addition to the consumption of information provided by the infrastructure through ALTO's Network Map and Cost Map interfaces, to request the creation of virtual links (Intents).

But more recent research, such as [6], it is noted that the Service Mesh approach is a strategy that addresses this issue more closely to current infrastructure orchestration patterns.

VI. Evaluation of the Topology Aware DNS

The Topology Aware DNS evaluation aims to identify the benefit of using this mechanism as a complementary strategy to the current service. Because until now, the mechanisms implemented are intended only for video distribution, using HTTP Redirect.

A. Experimentation scenario

To perform the tests, Topology Aware DNS was instantiated on two servers, defined as Authoritative DNS of a domain that was set just for this experiment. A test file was also placed in each of the servers positioned in each PoP of the infrastructure indicated in figure 1, which were registered in the Topology Aware DNS as available to answer the requests of this domain. Additionally, a call to this file was included in the eduplay page. Thus, each user that will access the service will request this file, going through the DNS query process.

The geolocation information of the IP address blocks was extracted from the free developer version of IP2Location[2]. For optimization purposes, a clustering process of consecutive address blocks belonging to the same country was performed, with the exception of Brazil, since the infrastructure nodes are located in this country. In this case, the clustering was performed based on the federal states that make up the country. Thus, after this processing, the 3,012,788 IPv4 address blocks were grouped into 281,868. Likewise, the 4,334,275 IPv6 address blocks were grouped into 570,646. The link map is based on the infrastructure topology, which contains 27 PoPs.

The data collection period was 30 days. 83,680 DNS type A (IPv4) requests were registered, coming from 63 countries. The total number of http requests was 35,629, also from 63 countries. The response time to requests was 244 microseconds in the 50th percentile, and 459 microseconds in the 99th percentile.

[2]https://lite.ip2location.com

B. Analysis of Results

The first set of data analysis was in characterizing the use of the ECS. It was identified that only 17% of the total DNS requests contained the client origin information (ECS). Of these requests, 94.55% came from Google's Recursive DNS. Of which 84.65% came from Brazilian clients. Another characteristic is that 4.91% of these requests come from networks with up to 256 addresses in its allocation block (class C). 79.90% are from clients that belong to networks that have between 256 and 65536, and 15.19% with networks higher than 65536.

Although the ECS requests do not represent a significant value, the nearest node selection accuracy factor was very positive. 41.46% of the http requests were forwarded to the closest server, and another 35.52% to servers in adjacent regions. Thus, the precision factor for 76.99% of the requests was optimum or good. Another 21.47% can be classified as average, and only 1.51% as poor (figure 2).

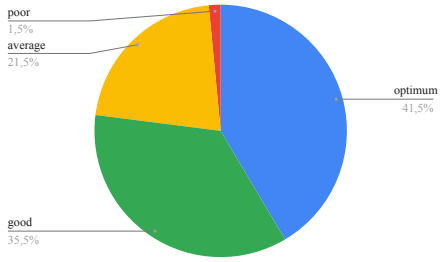

Fig. 2. Accuracy of node selection

To evaluate the advantage of this matter, a complementary test was performed to quantify the impact on content access from servers in different regions. In short, http requests were performed, downloading a 3 megabyte file, measuring the transfer rate. Subsequently, the download speed was compared, and associated access latency with the server. The results can be seen in the figure 3.

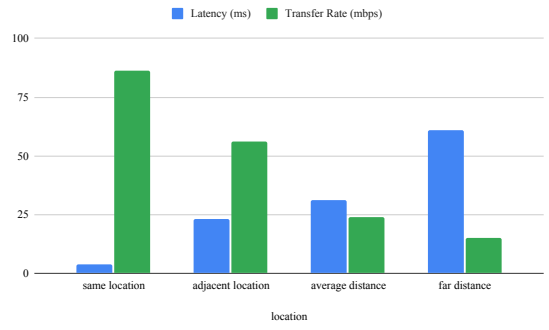

Fig. 3. Impact of latency on the transfer rate

The servers in the user's region have an average latency of 4ms, and achieved transfer rates of 86.4 mbps. When accessing the server in an adjacent network, the latency is 23ms and the rate is 53 mbps. At the server at an intermediate point, with

31 ms latency, the rate is 24 mbps. Finally, at the one further away, with latency of 61 ms, the rate is only 15.2 mbps.

VII. Related Work

The search for greater efficiency in edge computing has been the subject of many researches. One example is the work presented in [16]‘. This aims to minimize network latency in accessing services in Smart Cities using elements orchestrated in Kubernetes. For this, an extension to the standard Kubernetes Scheduler is proposed, in which it includes among the decision parameters, the measurements of RTT (Round Trip Time) and bandwidth between nodes. The results showed an 80% reduction in latency when accessing these services.

In this same vein, in [17] a strategy is proposed for the allocation scheduler aiming to address aspects of edge infrastructure. It defines two components: (i) the classifier, responsible for receiving all the scheduling requests and classifying them according to predefined metrics and (ii) the Edge Scheduler, responsible for selecting the available locations to deploy containers that are closest to the clients.

In parallel, it is noted that more recent efforts have been focused mainly on exploring the use of Machine Learning to analyze the various variables to be weighted in decision making. In [18] Deep Reinforcement Learning is used to search for the balance point between resource usage and maintaining Quality of Service in a data center. The results obtained showed significant gains in energy savings, improving efficiency by 47.88%.

VIII. Conclusions and Future Work

The great demand for digital services in the various segments, both in logistics as well as in service offering, has demanded the use of infrastructures with lower network latency. With this, the need for Edge Computing becomes more evident.

However, as highlighted in this paper, the efficient use of this kind of infrastructure has several challenges. Therefore, this paper aims to contribute to the identification of solutions that can address these challenges.

To achieve this, it presented Topology Aware DNS as an alternative to address the first indicated challenge **C1**. The main contribution of this paper is the proposal to address challenge **C2**. This is based on the use of 5 indicators that synthesize the information needed for an ML algorithm to manage the allocation of instances needed to meet the demand of a service offered on an Edge Computing. Furthermore, it also indicates possible routes proposed in other research to address the third challenge **C3**.

The evaluation of the approach used to address the first challenge was performed through a set of experiments conducted in a real environment. The results obtained showed that 76.99% of the requests were answered by nodes located in regions classified as optimal or good.

As a continuation of this work, the elements that make use of the indicators proposed in the section IV are being implemented to perform the allocation of the components efficiently.

References

[1] D. C. Uchoa, R. Kulesza, R. Matushima, S. Kopp, G. Bressan, and R. M. Silveira, “A management platform for multimedia distribution in country-wide networks,” in *2007 Latin American Network Operations and Management Symposium*, 2007, pp. 20–27.

[2] D. C. Uchôa, S. Kopp, H. M. Pimentel, R. Matushima, and R. M. Silveira, “An overlay application-layer multicast infrastructure,” in *2009 International Conference on Advanced Information Networking and Applications*, 2009, pp. 233–240.

[3] H. M. Pimentel, S. Kopp, M. A. Simplicio Jr., R. M. Silveira, and G. Bressan, “Ocp: A protocol for secure communication in federated content networks,” *Computer Communications*, vol. 68, pp. 47–60, 2015, security and Privacy in Unified Communications Challenges and Solutions.

[4] C. Contavalli, W. van der Gaast, D. C. Lawrence, and W. Kumari, “Rfc 7871-client subnet in dns queries,” 2016.

[5] E. Pujol, I. Poese, J. Zerwas, G. Smaragdakis, and A. Feldmann, “Steering hyper-giants’ traffic at scale,” in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 82–95.

[6] G. Antichi and G. Rétvári, “Full-stack sdn: The next big challenge?” in *Proceedings of the Symposium on SDN Research*, ser. SOSR ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 48–54.

[7] R. Bellman, “Adaptive control processes; a guided tour, princeton univ,” *Press, NJ*, 1961.

[8] R. S. Pressman *et al.*, “A practitioner’s approach,” *Software Engineering*, 2010.

[9] M. Amiri and L. Mohammad-Khanli, “Survey on prediction models of applications for resources provisioning in cloud,” *Journal of Network and Computer Applications*, vol. 82, pp. 93–113, 2017.

[10] M. Cheng, J. Li, and S. Nazarian, “Drl-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers,” in *23rd Asia and South Pacific Design Automation Conference, ASP-DAC 2018, Jeju, Korea (South), January 22-25, 2018*. IEEE, 2018, pp. 129–134.

[11] C. Bitsakos, I. Konstantinou, and N. Koziris, “Derp: A deep reinforcement learning cloud system for elastic resource provisioning,” *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 21–29, 2018.

[12] B. Du, C. Wu, and Z. Huang, “Learning resource allocation and pricing for cloud profit maximization,” in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI’19/IAAI’19/EAAI’19. AAAI Press, 2019.

[13] E. Durfee, “Blissiful ignoreance: Knowing just enough to coordinate well,” in *Proc. of ICMAS-95*, 1995, pp. 406–413.

[14] S. Kopp, M. P. Hernandez, L. T. Chigami, and R. M. Silveira, “Content delivery networks integrated with software defined networks,” in *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, ser. WebMedia ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 89–92.

[15] R. Alimi, R. Penno, Y. Yang, S. Kiesel, S. Previdi, W. Roome, S. Shalunov, and R. Woundy, “Application-layer traffic optimization (alto) protocol,” *RFC 7285*, 2014.

[16] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, “Towards network-aware resource provisioning in kubernetes for fog computing applications,” in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 351–359.

[17] W. Wong, A. Zavodovski, P. Zhou, and J. Kangasharju, “Container deployment strategy for edge networking,” in *Proceedings of the 4th Workshop on Middleware for Edge Clouds & Cloudlets*, 2019, pp. 1–6.

[18] M. Cheng, J. Li, P. Bogdan, and S. Nazarian, “H2o-cloud: A resource and quality of service-aware task scheduling framework for warehouse-scale data centers,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2925–2937, 2019.

[19] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, “Drl-scheduling: An intelligent qos-aware job scheduling framework for applications in clouds,” *IEEE Access*, vol. 6, pp. 55 112–55 125, 2018.