An Effective Resource Discovery Strategy for Fog Computing Driven by Computational Capabilities and Behavioral Characteristics

Joao Bachiega Jr.[1], Breno Costa[1] Leonardo R. Carvalho[1], Aleteia Araujo[1,2], and Rajkumar Buyya[2]

[1] University of Brasilia, Brazil
[2] The University of Melbourne, Australia
{joao.bachiega.jr, brenogcosta, leouesb}@gmail.com, aleteia@unb.br, rbuyya@unimelb.edu.au

Abstract. The fog computing paradigm allows for the distribution of computing resources and services at the edge of the network, close to end users, complementing cloud computing. Due to the dynamicity of fog computing environments, resource discovery is a key process that aims to find new computational resources that are available to integrate into it. These resources compose fog nodes, devices considering computational capabilities (such as CPU, memory, and disk) and behavioral characteristics (such as availability, scalability, and mobility). Performing an optimized resource discovery with all those attributes is still a challenge. This article proposes an efficient approach to resource discovery in fog computing that considers the computational capability and behavioral characteristics to select fog nodes. The results show that it is at least 33% more efficient than a similar solution found in the literature.

Keywords: resource discovery, fog computing, behavioral characteristics, computational capability

1 Introduction

Fog Computing has emerged as a promising solution to meet the growing demand to expand the capacity of computational, network, and storage resources closer to end users, compensating for some limitations of cloud Computing [17]. Among the current challenges, the effective discovery of computational resources is crucial. It occurs because in a fog computing environment characterized by the prevalence of dynamic contexts, such as the Internet of Things (IoT), and by competition for allocated computing resources, unpredictable events can arise, such as unavailability of services and devices, long response times, high latency, and reduced reliability [26]. To increase the challenge, applying resource discovery strategies from other consolidated computing paradigms, such as cloud computing, may present additional obstacles, as fog has computational resource constraints and more significant heterogeneity between devices. These points

highlight the urgency of developing new approaches and strategies to address the challenges in resource discovery, specifically in fog computing environments.

A fundamental concept in the fog computing environment is the fog node, defined as any device with software and hardware resources added to the high communication capability [3]. However, with a deeper understanding of the fog node, it is possible to observe that this node comprises computational capabilities, such as CPU, storage, memory, and any other hardware attributes that can be measured. In addition, reflecting the dynamicity and unpredictability of fog computing, the fog node is also made up of some behavioral characteristics, such as availability, interoperability, location awareness, and mobility [23].

Resource discovery refers to the search for available computational resources in a fog computing environment [4]. However, unlike other recent solutions in the literature [14, 25], this work presents a novel proposal for resource discovery that considers computational capabilities and behavioral characteristics in the discovery process. The strength of the proposed solution is that it consumes few computational resources in the discovery process. It will select and register only devices that meet all defined input criteria in the Resource Catalog. In addition, there is no need to install an agent previously on the fog node for discovery. With this, concern about computational resource limitations in the fog environment is fully addressed.

The remainder of the paper is organized as follows. Section 2 presents the main concepts of fog computing and contextualizes the discovery process as a part of the resource management process, highlighting its goals and challenges. The resource discovery proposal for fog, considering computational capabilities and behavioral characteristics, is presented in Section 3. A comparison with the main related works presented in the literature is made in Section 4. The configuration of the real environment used to conduct the experiments and the results obtained are presented in Section 5. Finally, Section 6 presents the conclusion of this work and the directions for future work.

2 Resource Management in Fog Computing

Fog computing is a computational paradigm that aims to extend the cloud computing model to be closer to the end devices, just as in nature, where fog occurs when a cloud approaches the ground. Therefore, it is characterized as a highly distributed computational model, enabling data processing at the edge of the network, but maintaining some integration with cloud computing. According to the National Institute of Standards and Technology (NIST) [11], the essential characteristics of fog computing are low latency, high geographical distribution, heterogeneity, interoperability, and scalability.

It is also important to note that fog computing is often confused with similar computational paradigms, like Edge, Mobile Cloud, Mist, or Dew Computing. In [5], a comparative analysis between fog computing and these related paradigms is presented. Therefore, according to [6], the fog computing concept is broader

and more complete and can be considered an umbrella that encompasses all these similar paradigms.

A key aspect of the fog computing architecture revolves around the fog node [11]. These nodes, whether physical elements or virtual components, establish a significant connection with end devices, providing computing resources. Unlike more mature computing paradigms, such as cloud computing, where computational instances are more homogeneous, located in specific datacenters, and controlled by a service provider, fog nodes are heterogeneous, have more limited computational capacity and network connections, and may be in movement. These different aspects should be considered to better utilize the fog nodes. Therefore, we assume that a fog node is composed of its computational capacity (i.e., CPU, memory, storage) and behavioral characteristics such as availability, security, and mobility. An in-depth computational analysis of a fog node can be found in [3].

In terms of architecture, a three-tier option is widely adopted to represent the fog computing infrastructure [18, 19]. The IoT layer represents the consolidation of all connected IoT devices at the edge of the network. End users can control these devices or autonomously collect environmental information to be forwarded for processing and storage in the Fog or Cloud layers. The Fog layer comprises fog nodes, and requests that do not require substantial computational resources can be served directly by this layer. Furthermore, the Fog layer plays a fundamental intermediary role between the IoT and Cloud layers, providing the necessary computational power to perform tasks such as filtering and aggregating data before forwarding them to the Cloud layer whenever there are insufficient resources available in the Fog layer [2]. On the other hand, the Cloud layer is characterized by more robust computational resources, and it is responsible for processing requests that could not be fully addressed by the Fog layer [2].

The fog nodes are managed by a process called resource management, which consists of five distinct steps [4] briefly explained below:

1. **Discovery**: aims to find new resources in the dynamic fog environment, making them available for use after updating the Resource Catalog [4];
2. **Estimation**: is the calculation of the amount of computing resources and time that are needed to perform a set of tasks [4];
3. **Allocation**: aims to select the best-cataloged resources to execute a workload in the fog environment, according to the users' criteria [4];
4. **Monitoring**: collects status data about fog nodes, their resources, and communication links, maintaining updated the Resource Catalog, and providing relevant information to the orchestrator for decision making [7, 9];
5. **Orchestration**: is a management function responsible for the service life cycle that provides requested services to the user and assures the Service Level Agreement (SLA) [8].

Each step plays a specific role in optimizing the use of available resources in the fog infrastructure, contributing to the efficient and reliable performance of the applications and services offered in this environment. Once the discovery

process has successfully registered the fog node in the catalog, its subsequent monitoring occurs in the monitoring and orchestration steps. Therefore, the resource discovery solution must be closely aligned with other resource management processes, ensuring that the entry criteria are met and expected results are generated, thus contributing to effective management [8].

In this context, the resource discovery procedure plays an essential role in fog computing resource management, allowing the identification and selection of new computational resources for future task submissions. This process may involve creating a list of resources for subsequent selection [24]. At its core, resource discovery encompasses activities related to locating and disseminating information about resources, making it essential to fully exploit all distributed resources in the fog environment [27].

Furthermore, it is important to note that resource discovery specifically for a fog computing environment has some requirements and challenges that must be addressed properly. Among them, the following are worth highlighting:

1. **Location**: the solution should be situated near the resources, ensuring efficiency in resource allocation [7];
2. **Availability of Resources**: it is essential to account for the intermittent availability of fog nodes and the potential for them to be powered off at specific times [21];
3. **Fault Tolerance**: the creation of fault-tolerant solutions is vital because services are distributed across multiple nodes [7];
4. **Asynchronous Notification**: it is a key strategy to address information synchronization issues and communication failures [15];
5. **Different Communication Protocols and Resource Types**: need for flexibility regarding identification schemes, enabling discovery regardless of the communication technologies and protocols adopted by the fog nodes, as well as their heterogeneity in terms of resource types [15];
6. **Lightness**: to mitigate substantial performance impacts, the solution should be lightweight, aligning with the limited resources of fog nodes [21].

Effectively addressing fog computing's unique characteristics compels resource discovery solutions to fulfill novel requirements that differ from those in established computational paradigms. This underscores the need for specific approaches, such as the one presented in the next section.

3 Resource Discovery Proposal

In resource discovery in a fog computing environment, the central objective is to identify and map the resources available in the environment efficiently and dynamically. This step is essential to ensure that applications can access the necessary resources in a timely and optimized manner.

3.1 Problem Definition

In a fog computing environment, there is a set of fog nodes, where each $f_i \in \mathcal{F} = \{f_1, f_2, ..., f_F\}$. The attributes of computational capabilities ($\mathcal{C}$) represent minimal computational requirements for task execution, such as the number of CPUs, the amount of memory, disk, etc. They are represented by the vector $\mathcal{C} = \{c_1, c_2, ..., c_C\}$, where C is the cardinality of the set. In contrast, behavioral characteristics ($\mathcal{B}$) can significantly impact user experience and application performance but do not impede task execution. Security, mobility, scalability, and reliability are examples of behavioral characteristics. They are represented by $\mathcal{B} = \{b_1, b_2, ..., b_B\}$, where B is the cardinality of the set. Taking into account both types of attributes (computational and behavioral), each fog node is determined as a tuple $f_i = \{\mathcal{C}_i \cup \mathcal{B}_i\}$; i.e., $f_i = \{c_{i1}, ..c_{iC}, b_{i1}, ..., b_{iB}\} \mid i \in \{1, 2, ..., F\}$.

With this, considering that each column represents an attribute (a, either computational or behavioral), the Resource Catalog can be denoted by an attribute matrix (A), where $\{a_{in} \in A \mid 0 < i \leq F; 0 < n \leq C + B\}$. Therefore, the Resource Catalog can be represented as:

$$A = \begin{vmatrix} c_{11} & c_{12} & ... & c_{1C} & b_{11} & b_{12} & ... & b_{1B} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{F1} & c_{F2} & ... & c_{FC} & b_{F1} & b_{F2} & ... & b_{FB} \end{vmatrix}$$

The goal of resource discovery in fog computing is to find new available fog nodes in the environment and record their computational capabilities ($\mathcal{C}$) and behavior characteristics ($\mathcal{B}$) in the Resource Catalog (A).

Finally, a vector $\mathcal{V} = \{v_1, v_2, ..., v_{C+B}\}$ brings the minimum values required for each of the attributes. It is used to effectively select and register only fog nodes that meet the computational and behavioral criteria in the Resource Catalog. Therefore, the objective function can be defined as presented in Equation 1, for each $f_i \in \mathcal{F}$:

$$\mathcal{O}_i = \sum_{n=1}^{C+B} (a_{in} \geq v_n) \tag{1}$$

In other words, for each fog node found, the variable $\mathcal{O}$ increases with each criterion met (an attribute equal to or greater than its defined minimum value), and the registration in the Resource Catalog will only occur if the attributes respect the defined baseline $\mathcal{V}$. Therefore, the restrictions for this problem are that all needed computational capabilities ($\mathcal{C}$) and behavior characteristics ($\mathcal{B}$) must be provided by the fog node being evaluated after discovery. The total value of $\mathcal{O}$ for each fog node must be the sum of the number of attributes indicated by Equation 2.

$$\mathcal{O}_i = C + B \tag{2}$$

3.2 Proposal

Considering the essential characteristics of fog computing and the resource discovery process (Section 2), this section presents a novel proposal for discovery that brings into focus computational capabilities and behavioral characteristics when selecting fog nodes. In this context, the following parameters were used:

1. **CPU:** the number of CPU cores available to be exclusively used must be equal to or greater than the number of the CPU requested by the user;
2. **CPU Clock:** the frequency at which a processor's clock can generate pulses, which are used to synchronize the operations of its components. It is important to define the processor´s speed, and it´s measured in Megahertz (Mhz);
3. **Memory:** the minimum requirement for RAM (in GB) available for exclusive use in the fog node is the amount of RAM asked for by the user;
4. **Storage:** the amount of free storage (in GB) available for exclusive use in the fog node must be at least the amount of free storage asked for by the user;
5. **Latency:** the delay (in milliseconds) for a request to be transferred must be equal to or less than the amount of delay asked for by the user;
6. **Number of hops:** define the locality and estimate the distance between the fog node and the server, making it easier to find, for example, the nearest fog nodes;
7. **Availability:** considering historical behavior, how much of the resource is available for allocation. Using the AWS SLA[3] as a point of comparison, when the fog node availability is higher than 99%, it has the value 5. For availability lower than 95%, the value 1 is assigned;
8. **Scalability:** the ability of the fog node to handle an increasing workload, considering both historical behavior and the amount of free resources. The value 5 is assigned to resources capable of expanding their computing power by up to 10 times. The value 4 is assigned to those with 8 times, and successively until value 1;
9. **Reliability:** the level at which the fog node is trusted to complete task execution, considering historical behavior. The value 1 is assigned to the fog nodes that always abort an execution. On the other side, the value 5 is assigned to the fog nodes that, in at least the last 10 times, have not aborted an execution;
10. **Mobility:** how mobile the fog node is. It is a proportionally inverse metric, where the more static a fog node is, the higher the mobility value. In [1], a scale to classify the device´s mobility is presented, assigning different values to large mobile devices (such as tablets and laptops), small mobile devices (i.e., smartphones), and also static devices (i.e., desktops). This scale was adapted to determine the values between 1 and 5 to be assigned in this attribute.

The input parameters are the range of IPs to be searched and the computational and behavioral criteria. The process is initiated on a centralized server by

[3] https://aws.amazon.com/compute/sla/

searching for the hosts in the network indicated in the entry using the NMAP protocol [20]. This protocol was chosen for this task because it can already provide relevant information for other algorithm decisions, such as latency and the MAC Address of the fog node found. The fog node found is compared with the records in the Resource Catalog through the MAC Address. If it is an unknown fog node, the process continues to collect information about the computational parameters. To obtain this, the host is accessed via the SSH protocol.

Once the values of the computational capabilities and behavioral characteristics have been obtained, validation is performed. Only if all attributes meet the requested criteria are recorded in the Resource Catalog, which will be available for use by other resource management processes, such as allocation and monitoring. For the proposal, the Resource Catalog is a file in JSON format, as it is understood that this type of file favors the exchange of messages and is commonly used in different applications.

A recurrence time is provided to restart the process to guarantee the dynamicity required in a fog environment. However, changing its bootstrapping method from proactive to reactive is possible with only a small adjustment to the algorithm that supports the proposal. As a result, scanning for new hosts would only occur when requested, allowing the discovery server's network and computational resource consumption overhead to be further reduced.

One of the novelties of this proposal is that it allows one to filter the fog nodes of interest by computational and behavioral criteria. If a fog node does not meet the minimum criteria provided right after it has been discovered, it is ignored and not registered in the Resource Catalog. Another point worth highlighting about this proposal is that it does not require the previous installation of an agent on the fog node. This is important because the computational resources of a fog environment tend to have a low capacity, and the installation of an agent could have a negative impact.

4 Related Work

Although the literature on resource discovery contains some solutions, no article explicitly related to fog computing provided an analysis of both computational capabilities and behavioral characteristics in the discovery process. Few articles detailed all the experiments, and fewer were tested in real fog computing environments. This section presents the most important work in this area.

In the study provided by Jin and Kim [13], a RESTful API modeling language is used to articulate resources, and an Open Communication Foundation (OCF) specification model protocol is used to facilitate communication between fog nodes. A Resource Catalog was used on the Resource Discovery server to store information, which can be later queried during subsequent resource management processes. The experiments were conducted in a real environment using only two devices. There is no analysis or comparison of the solution proposed by the authors with other similar articles.

In the paper by Sattari et al. [22], a distributed resource discovery solution for mist computing platforms is suggested, adopting the Constrained RESTful Environments (CoRE) Resource Directory. Evaluations include simulated and real-world scenarios in an environment with two Raspberry Pis as fog devices and an undefined number of ESP8266 embedded devices as Mist devices. Because the experiments carried out by the authors focused on latency and power consumption, there is no possibility of comparing the results obtained in our work with those they presented.

In [16], the Named Data Networking (NDN) protocol highlights its suitability for resource discovery in IoT and fog computing environments. This protocol provides information about the resource, eliminating the need to overload the architecture to obtain these data and, consequently, reducing the time required to locate the most appropriate node. The experiments were performed based on four metrics: acceptance ratio, overhead, number of hops, and routing cost, which cannot be used to compare with our work.

To support a specific healthcare scenario, the authors of [12] employ a heuristic algorithm to achieve locally optimal results in resource discovery, considering latency as a crucial quality of service requirement. The centralized architecture is distributed between the cloud, fog, and IoT layers, and the proposal is based on a reactive search that requires an agent across the fog nodes. There is no analysis or comparison of the solution proposed by the authors with other similar articles. The matching function's performance evaluation and resource efficiency were assessed by comparing the containerized and non-containerized implementations developed by the authors.

Finally, the paper [25] characterizes resources through WSDL, along with a metaheuristic algorithm that searches for fog nodes, considering response time, fitness function, and cost criteria. The paper was the only one in this section that compared the results with other solutions (Grey Wolf Optimizer and Genetic Algorithm). However, comparing our metrics with those in their article is impossible once they consider different metrics and approaches.

Table 1. Analysis of related work.

Paper	Year	Environment	Computational	Behavioral	Catalog	Experiments	Devices	Comparison
[13]	2018	IoT	Yes	No	Yes	Real	2	No
[22]	2020	Mist	Yes	No	No	Real	2	No
[16]	2021	Edge	Yes	No	No	Simulation	–	No
[12]	2021	Edge	Yes	No	No	Real	7	No
[25]	2022	Edge	Yes	No	No	Simulation	–	Yes
This work	2024	Fog	Yes	Yes	Yes	Real	32	Yes

Table 1 summarizes the related work analyzed in this section. It is possible to identify that none of the works presented gave resource discovery solutions specifically for fog computing, although Edge and Mist computing are also distributed paradigms. Identifying a deficiency in performing experiments in real environments is also possible. The ones that evaluated their works on real-world

testbeds [12,13,22] have used several devices much lower than those used in this work. It is also possible to observe the absence of proposals considering computational capabilities and behavioral characteristics and register the result in a Resource Catalog. Another highlight of this study is that it is one of the few studies that compare the results obtained with any other work available in the literature.

5 Performance Evaluation

A real experiment environment is set up to support the evaluation of the proposal, simulating the expected parameters and behaviors. This environment consisted of different types of IoT devices with four Raspberry Pi units, each with 4GB of RAM, 2GB of disk size, and 4 CPUs; and four virtual machines with 8GB of RAM, 4GB of disk size, and 2 vCPUs were also added to the environment to carry out the experiments. The server was a 16GB RAM quad-core device. In total, the real experiment environment consisted of 32 devices. The IoT devices and the server were in the same network range, based on a wireless connection. The number of hops between the server and the fog nodes was the same for all experiments performed to ensure fair execution and the proximity and low latency expected from a fog computing environment.

For the scope of this work, the proposed algorithm can bring information from fog nodes that use the Linux operating system. A previous bootstrap process was required on each host, providing the security credentials needed to execute the Linux commands by the discovery application and improving security and privacy. In this way, the operating system is also considered a criterion of the computational capacity to be met. Still, with the same objective of increasing security, the default port of the SSH service was changed. As all commands are considered native, installing additional host packages was unnecessary. On the server side, it was necessary to install the NMAP package[4].

5.1 Experiments and Results

This section presents the experiments and results obtained by the algorithm proposed for resource discovery in a real fog computing environment.

Experiment 1 - Completeness of the Search

The first experiment concerned the algorithm's ability to find all available and eligible fog nodes in the environment. For this experiment, the computational and behavioral criteria were configured lower than the existing capabilities of the testbed's devices to allow discovery, and the required operating system was set to Linux. All fog nodes were turned on to do this, and the discovery algorithm started. The algorithm could discover all fog nodes that met the requirements and bring all the data discovered to the Resource Catalog.

[4] https://NMAP.org/download#linux-rpm

It is important to note that after all fog nodes are found in the first run, the next runs will not bring new results until other new fog nodes become available in the network. Another possibility is that one or more fog nodes be removed from the Resource Catalog (i.e., by monitoring process due to timeouts on heartbeat messages). This shows that the algorithm can compare the fog nodes found with those already registered in the Resource Catalog and avoid duplication of records, data inconsistency, and computational effort for the unnecessary registration of a fog node that was already registered. To guarantee this, the fog node's MAC address is considered the primary key.

This experiment was also carried out by starting the discovery algorithm without any fog nodes connected and connecting them one by one, with intervals of 2 minutes between them, still with the computational and behavioral criteria lower than those existing on the hosts. For this step, the algorithm recurrence time was set to 1 minute. After the execution of the experiment, it was possible to confirm that the fog nodes are discovered on the network as soon as they are connected.

Experiment 2 - Computational Capability Constraints

To continue testing, the *memory* criterion was adjusted to 3000 MB, filtering the discovery only to fog nodes with free memory greater than that number. Table 2 presents the free memory value for each of the fog nodes that are Raspberry Pis and Virtual Machines at the time of the experiments.

Table 2. Free memory values.

Fog Node	**FN01**	**FN02**	**FN03**	**FN04**	**FN05**	**FN06**	**FN07**	**FN08**
Free Memory (MB)	3569	3123	2681	2983	6852	6230	5877	6541

In the first run, only two fog nodes would not meet the selection criteria, and only fog nodes FN03 and FN04 were not found. In a second execution, the criterion was adjusted to 3500 MB, and again, the algorithm was assertive, resulting in the additional exclusion of FN02.

This functionality of the algorithm is also novel, and it is considered important to guarantee the search for fog nodes that are viable for the use case, avoiding unnecessary registrations in the Resource Catalog, and optimizing the registration time for the discovery and search process for other stages of resource management that will use the Resource Catalog.

Experiment 3 - Runtime

Another experiment was carried out to measure the execution time concerning the number of fog nodes to be discovered. However, with the execution of experiments, it was realized that a factor that impacts the algorithm's execution time is the range of network IPs being searched. The network range is a parameter defined as input to the algorithm, as indicated in Section 3. When considering the range 192.168.2.0/28 as input, for example, 13 IPs will be verified, from 192.168.2.2 to 192.168.2.14, since 3 IPs are used for network (192.168.2.0),

gateway (192.168.2.1), and broadcast (192.168.2.15). With this, in the range 192.168.2.0/28, 1021 IPs will be verified.

However, the growth is not linear with the number of IPs verified. While a range /28 is verified in 3 seconds, a /22 needs 38 seconds. This is a point to note, as a fog computing environment is expected to have many different networks and, consequently, an extensive range of IPs to be checked. One way to solve this situation was to run the algorithm in parallel, limiting the IP range to a reasonable time for the solution.

To this end, the IPs of some devices were adjusted to be on four different subnets. The script that contains the parallelized calls to the algorithm had an average execution time of 22 seconds and, in the end, was able to find all existing fog nodes. For this experiment, the network range parameter was set to /24 in 4 tasks, each searching in a subnet.

Suppose this action of parallelization executions in limited network ranges had not been done. The average time required to discover the fog nodes in 4 different subnets would have been 41 seconds, almost twice as long. In that case, a /22 range in the model without changes to the algorithm would have taken almost twice as long.

Another situation that had an even more significant impact on execution time occurred. To simulate a fog computing environment with high heterogeneity of resources even more accurately, several devices not eligible or not choosing to participate in the fog environment were registered on the same WiFi network and the same subnet. These devices include personal cell phones, routers, smart TVs, etc. In this scenario, it was possible to identify that after the *NMAP* scan finds any device, an attempt is made to connect to it to extract its computational information using the *ssh* Linux command. This process requires time until the destination host denies the connection, causing slower execution on network ranges with many devices not eligible as fog nodes.

To solve this situation, the NMAP command search process was improved to return only devices with a specific open port. This port is configured on the fog node during the bootstrapping process. Therefore, the list of hosts found is limited to potential candidates belonging to the fog computing environment.

Table 3 presents the execution time of the algorithm before and after the implementation of the improvement. The number of devices available in the IP range and how many were eligible to belong to the fog environment are also shown. By analyzing Table 3, it is possible to observe a significant reduction in algorithm execution time in the first two IP ranges, in which some devices were not fog nodes eligible for discovery. For the last two IP ranges, there was no change in execution time as no devices were available. With this, it was possible to identify that the change made to the NMAP search parameter was sufficient to improve the performance of the algorithm's execution time.

Experiment 4 - Network Consumption

Finally, the network consumption of the discovery algorithm was considered. This concern is relevant in a fog environment potentially connected by unstable connections caused by multiple technologies and device mobility. For this mea-

Table 3. Runtime in testbed environment.

IP Range	Total Devices	Possible Fog Nodes	Old Runtime (seconds)	New Runtime (seconds)
192.168.0.1-64	29	8	1032	43
192.168.0.65-128	3	0	364	12
192.168.0.129-192	0	0	3	3
192.168.0.193-254	0	0	3	3

sure, the execution of the command `ifstat -t` was used on the server, which shows the incoming and outgoing packets over time[5].

Both input and output flows have similar behavior. Network consumption increases when the first fog node starts reporting on the network, remaining constant even with new fog nodes added over time. Thus, network consumption is unaffected by the number of fog nodes discovered and available.

Throughout the 716-second collection window, the maximum input consumption was 11.36 KB/s, with an average of 2.55 KB/s of consumption. Regarding output, the peak was 7.91 KB/s, and the average was 1.94 KB/s, demonstrating that the proposed algorithm demands little network packet output on the server side.

6 Conclusions and Future Work

The discovery of resources in a fog computing environment requires proposing solutions capable of dealing with the heterogeneity and high geographic distribution that this new computational paradigm presents as essential characteristics. In addition, solutions must be lightweight to not burden either the computational capacity of its components or the network.

This paper proposed a novel computational capabilities and behavioral characteristics resource discovery algorithm for fog computing. It is based on a network-wide search for devices eligible to be fog nodes, and these are added to a Resource Catalog only if they meet the defined computational and behavioral criteria.

Through experiments in a real environment, it was possible to identify that the proposed solution efficiently finds fog nodes available in the environment and registers them in the Resource Catalog with little computational effort and low network consumption.

The use of the real environment for experiments stands out, as it was possible to detect several improvements in the algorithm. These improvements, such as optimizing information exchange between protocols and network equipment, including fog nodes with other operating systems and architectures, and experiments with more devices, will be treated as future work. Furthermore, we plan to implement the proposed discovery algorithm in fog computing software frameworks such as FogBus2 [10] for end-to-end evaluation in a real fog computing

[5] https://linux.die.net/man/1/ifstat

environment for different classes of IoT applications. We also plan to explore application security and trust issues in the fog environment, which is critical for sensitive IoT applications.

Acknowledgments

This study was partially financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Brasil - Finance Code 001. Gratitude also goes to the BioCloud2 project, approved under CNPq/AWS Call No. 064/2022, CNPq process No. 421828/2022-6.

References

1. Aazam, M., Huh, E.N.: Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. Proceedings - International Conference on Advanced Information Networking and Applications, AINA 2015-April(March), 687–694 (2015)
2. Al-Doghman, F., Chaczko, Z., Ajayan, A.R., Klempous, R.: A review on fog computing technology. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 001525–001530 (Oct 2016)
3. Bachiega, J., da Costa, B.G.S., Araujo, A.P.F.: Computational perspective of the fog node. 22nd International Conference on Internet Computing and IoT (2021)
4. Bachiega Jr, J., Costa, B., Carvalho, L.R., Rosa, M.J., Araujo, A.: Computational resource allocation in fog computing: A comprehensive survey. ACM Computing Surveys 55(14s), 1–31 (2023)
5. Bachiega Jr., J., Costa, B., Carvalho, L., Oliveira, V.H., Santos, W., de Castro, M.C.S., Araujo, A.: From the sky to the ground: Comparing fog computing with related distributed paradigms. In: Proceedings of the 12th International Conference on Cloud Computing and Services Science (CLOSER 2022). pp. 158–169 (2022)
6. Chiang, M., Ha, S., Risso, F., Zhang, T., Chih-Lin, I.: Clarifying fog computing and networking: 10 questions and answers. IEEE Communications Magazine 55(4), 18–20 (2017)
7. Costa, B., Bachiega Jr, J., Carvalho, L.R., Rosa, M., Araujo, A.: Monitoring fog computing: A review, taxonomy and open challenges. Computer Networks 215, 109189 (2022)
8. Costa, B., Bachiega Jr, J., De Carvalho, L.R., Araujo, A.P.: Orchestration in fog computing: A comprehensive survey. ACM Computing Surveys (CSUR) 55(2), 1–34 (2022)
9. Costa, B., Banerjee, A., Jayaraman, P.P., Carvalho, L.R., Bachiega, J., Araujo, A.: Achieving Observability on Fog Computing with the Use of Open-Source Tools. Springer Nature Switzerland pp. 319 – 340 (2024)
10. Deng, Q., Goudarzi, M., Buyya, R.: Fogbus2: a lightweight and distributed container-based framework for integration of iot-enabled systems with edge and cloud computing. In: Proceedings of the International Workshop on Big Data in Emergent Distributed Environments. pp. 1–8 (2021)
11. Iorga, M., Feldman, L., Barton, R., Martin, M., Goren, N., Mahmoudi, C.: The NIST definition of fog computing. Tech. rep., National Institute of Standards and Technology (2018)

12. Islam, J., Kumar, T., Kovacevic, I., Harjula, E.: Resource-aware dynamic service deployment for local IoT edge computing: Healthcare use case. IEEE Access 9, 115868–115884 (2021)
13. Jin, W., Kim, D.: Consistent registration and discovery scheme for devices and web service providers based on RAML using embedded RD in OCF IoT network. Sustainability 10(12), 4706 (2018)
14. Karagiannis, V., Desai, N., Schulte, S., Punnekkat, S.: Addressing the node discovery problem in fog computing. In: 2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020). Schloss-Dagstuhl-Leibniz Zentrum für Informatik (2020)
15. Khalil, K., Elgazzar, K., Seliem, M., Bayoumi, M.: Resource discovery techniques in the internet of things: a review. Internet of Things 12, 100293 (2020)
16. Kondo, D., Ansquer, T., Tanigawa, Y., Tode, H.: Resource discovery for edge computing over named data networking. In: 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC). pp. 552–559. IEEE (2021)
17. Laghari, A.A., Jumani, A.K., Laghari, R.A.: Review and state of art of fog computing. Archives of Computational Methods in Engineering 28(5), 3631–3643 (2021)
18. Mahmud, M., Buyya, R.: Fog computing: A taxonomy, survey and future directions. Internet of Everything - Algorithms, Methodologies, Technologies and Perspectives (2016)
19. Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R.H., Morrow, M.J., Polakos, P.A.: A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. IEEE Communications Surveys and Tutorials 20(1), 416–464 (2018)
20. Orebaugh, A., Pinkard, B.: NMAP in the enterprise: your guide to network scanning. Elsevier (2011)
21. Pourghebleh, B., Hayyolalam, V., Aghaei Anvigh, A.: Service discovery in the internet of things: review of current trends and research challenges. Wireless Networks 26(7), 5371–5391 (2020)
22. Sattari, A., Ehsani, R., Leppänen, T., Pirttikangas, S., Riekki, J.: Edge-supported microservice-based resource discovery for mist computing. In: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress. pp. 462–468. IEEE (2020)
23. Sengupta, S., Garcia, J., Masip-Bruin, X.: Essentiality of resource and service-task characterization in the coordinated fog-to-cloud paradigm. In: 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT). pp. 249–254. IEEE (2018)
24. Singh, S., Chana, I.: Cloud resource provisioning: survey, status and future research directions. Knowledge and Information Systems 49(3), 1005–1069 (2016)
25. Wang, R., Lu, J.: QoS-aware service discovery and selection management for cloud-edge computing using a hybrid meta-heuristic algorithm in IoT. Wireless Personal Communications 126(3), 2269–2282 (2022)
26. Wang, Z., Goudarzi, M., Gong, M., Buyya, R.: Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments. Future Generation Computer Systems 152, 55–69 (2024)
27. Zarrin, J., Aguiar, R.L., Barraca, J.P.: Resource discovery for distributed computing systems: A comprehensive survey. Journal of parallel and distributed computing 113, 127–166 (2018)