2019 IEEE International Conference on Edge Computing (EDGE)

Decentralized Resource Auctioning for Latency-Sensitive Edge Computing

Cosmin Avasalcai
Distributed Systems Group
TU Wien
Vienna, Austria

Christos Tsigkanos
Distributed Systems Group
TU Wien
Vienna, Austria

Schahram Dustdar
Distributed Systems Group
TU Wien
Vienna, Austria

***Abstract*—The prevalence of Internet of Things (IoT) in contemporary settings has induced systems composed of heterogeneous devices, computing infrastructures, and cloud services. New paradigms have emerged where computational resources are managed closer to IoT end-devices, within a general theme of decoupling from the cloud. This is because meeting application demands must occur at runtime, in the face of uncertainty and in a decentralized manner. Taking advantage of available resources closer to devices calls for novel resource allocation techniques that comply with latency, privacy and decentralization demands of IoT applications. To this end, we propose a novel decentralized resource management technique and accompanying technical framework for the deployment of latency-sensitive IoT applications on edge devices. Our technique is inspired from the functionality of an auction house and has two objectives; (i) find a deployment mapping for an arbitrary application, compliant with its individual resource requirements and latency constraints, (ii) facilitate privacy, as each device participates at their own will, based on its own availability and privacy preferences. Our approach ensures seamless deployment at runtime, assuming no design-time knowledge of device resources or network topology.**

***Keywords*-Internet of Things; Edge Computing; Resource Management; Decentralization; Fog Computing;**

I. Introduction

With the advancement of Internet of Things (IoT) devices and network capabilities, new applications with more stringent requirements have emerged, requiring low end-to-end (e2e) latency and data privacy. The current state of the art in IoT is a centralized infrastructure where data, processing or control is situated on the cloud. However, the cloud-IoT coupling is not viable anymore due to its high latency, its being a central point of failure, and difficulties posing to privacy and other requirements. Moreover, due to the high volume of data generated by end-devices, there is an increased risk of congestion and bandwidth waste [1].

As a solution to such challenges, edge paradigms aim at moving computational, control or data resources from the cloud closer to the edge of the network. By enabling more processing capabilities at the edge, more responsive IoT applications are obtained with low e2e latencies and fast response times. Additionally, scalability is augmented since most of the collected data is processed at the edge and only some information is sent to the cloud. Privacy, a growing important goal, is obtained by processing private data locally [2]. However, to take advantage of these new available computational resources, novel resource management techniques are needed to manage the distributed resources appropriately.

Resource management [3] concerns controling resources' utilization in a heterogeneous IoT network where edge devices have limited computational resources and high uncertainty exists due to the dynamic nature of the IoT. Such uncertainty is introduced by (i) device heterogeneity, (ii) mobility, as devices like smart cars and smartphones make up the collective, and (iii) lack of knowledge at design time of the operational configuration of the system. Resource management has five different objectives; estimation, discovery, allocation, sharing, and optimization. In this paper, we propose a novel technique combining resource allocation and resource sharing for task allocation. Task allocation at the edge of the network has focused on offloading tasks from constraint mobile devices to save energy consumption [4] or relied on performing processing data and analytics in the Cloud [5]. However, such a design is not suitable for latency-sensitive application requirements and is unsuccessful in preserving the privacy of user's personal data.

In this paper, we propose a novel decentralized resource management framework with the purpose of deploying IoT applications at the edge of the network based on a set of objectives. We consider as objectives the satisfiability of some latency Service Level Agreement (SLA) and ensure data privacy requirements. We treat privacy as respecting resource preferences of each edge device, meaning that their resources are not advertised system-wide. Specifically, our contributions are as follows:

- A novel task allocation technique to share resources with nearby nodes based on application requirements;
- Participating nodes on the network utilize multiple bidding strategies, making their own choices regarding their local available resources;
- We advocate decentralization, as the system can operate without a stable connection to the cloud if there are enough available resources at the edge.

II. Decentralized Resource Auctioning

Our technical framework tackles decentralized resource management at runtime, focusing on latency-sensitive IoT applications. Its functionality is inspired by an *auction* *house*, reflecting core ideas of decentralization and device-participation inherent in our approach. Once an application arrives and is ready to be deployed, a list of auction participant devices called the *bidder nodes* is created. The participants share a direct communication link to the deployment node, which is called *the dispatcher*.

The dispatcher serves as the auction house owner for every individual deployed IoT application, and advertises the application that it desires to deploy on the network. Subsequently, the bidders, according to their own preferences and current state of free computational resources, bid for parts of the application, called *tasks*, communicating to the dispatcher their bids. The dispatcher, after collecting bids finds an allocation of tasks to the bidding nodes that satisfies given latency and other application requirements and deploys the application on the node collective.

Motivational example. As a simple scenario serving as a running exemplar, consider a public safety IoT application that helps the authorities in solving missing person cases faster. The application is based on video and image analysis and uses available resources taken from smartphones, CCTV and dashboards cameras found in the area of the incident, to locate a missing person. Specifically, the application is composed of components responsible for specific tasks, including (i) image analysis, (ii) face recognition, (iii) motion detection, and (iv) results generation.

Significant challenges emerge when deployment of such an application takes place on the cloud. First, privacy issues arise since collected data from user end-devices must be sent to the cloud for further analysis. Second, in such a use case time is valuable, hence the application requires fast response times and availability even when cloud connectivity cannot be established. As such, deployment close to the end-devices is desired; with application operation right at the incident location and distributed among participating devices.

III. Problem formulation

In edge computing, an application may utilize resources distributed among various connected devices. In our conception, an application is composed of a set of interconnected tasks, each of which may have resource requirements that need to be met by some edge node upon deployment.

A. IoT Application and System Model

IoT applications may be deployed across different connected devices; an application is assumed to be decomposed in distinct computational blocks, called *tasks*. A task has minimal requirements upon others, and requires certain *resources* to execute. We assume that each application contains a starting task which provides the required input data (i.e., a sensing task) and an end task to take action on the obtained results (i.e., an actuating task). This is in line with the fact of limited computational resources found at the edge of the network, where an application model reflects a division of the entire application functionality in multiple tasks, since the execution of the whole application in a single device may be infeasible or impractical [6].

A division of an application into tasks offers the possibility of better utilization of distributed resources from devices participating in the network. More concretely, we assume that an application is described by a set of tasks T=$\{t_1, t_2, ..., t_n\}$. Furthermore, since a workflow between tasks exists, the application is modeled as a directed acyclic graph (DAG), G_{app} = (V, E), where vertices represent the tasks and edges capture dependencies between them [4]. Considering this, we can model the IoT application from our motivational example as illustrated in Fig. 1 .

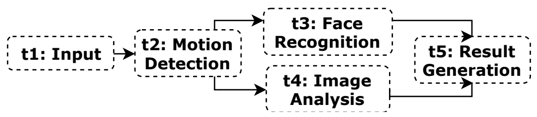

Figure 1. Model of the public safety IoT application.

A task t_i represents a set of instructions performing some specific application feature; computational or other resources may be required to execute the task. To this end, we assume each task specification to include a set of resource requirements $R_t=\{r_1, r_2, ..., r_n\}$ – these may reflect for instance generic memory, storage or computational aspects required to execute the task. However, resource requirements may specify particular data that the task requires to be present, or particular and domain-specific hardware, sensor or actuators. For example, besides computational resources, the *motion detection task* requires as input video data gathered from the proximity of each device.

The edge computing architecture we consider consists of multiple devices connected in a peer-to-peer manner. Devices have software stacks able to execute program code, and each device at any point in time has certain available resources $R_E=\{r_1, r_2, ..., r_n\}$. In essence, a correspondence and allocation of task resources to device resources is necessary to be established to execute tasks of an IoT application. Let EN=$E_1, E_2, ..., E_n$ denote all the devices available in some local setting, e.g. they may be placed in the same neighborhood or domain, such as a building or manufacturing floor. A device represents an edge node which may gather information from its environment, e.g., smartphones or small servers. We assume that every edge node trusts all devices to which establishes a direct communication link; they all belong to the same local administrative domain.

An edge $l_{i,j}$ describes the latency between E_i and E_j. Each node has an associated latency while communicating with its neighbors. As a result, when a task t_i is deployed on an edge device E_i and t_j is mapped on node E_j, the communication latency between the two tasks is inherited from the two edge nodes hosting them. Since knowing this latency is imperative to the overall mapping strategy, we assume that the latency is already provided by latency monitoring in place.We consider implementation of the latency monitoring functionality as out of scope of the present paper; we assume latencies between tasks are adequately monitored and provided.

B. *Objectives*

Latency is a prime concern for IoT applications and has been one of the main drivers for edge computing; we are concerned with a particular manifestation of latency, which is the *e2e delay* of an application's execution when operational. We define the e2e delay as the duration of time for an instance of the IoT application to be processed and yield a result. Moreover, privacy plays an important role in pervasive systems such the IoT-populated, edge computing ones we target, so we consider privacy an orthogonal, secondary objective that should always be satisfied by an application deployment. This is often the case since personal data found on edge devices should be especially protected when resources are shared. To achieve these privacy requirements, we ensure that the data is processed locally, on the device where it resides. In the end, we guarantee that the personal data is processed in the proximity of the user, by mapping tasks to the edge device containing the data.

Taking into account the two aforementioned objectives, a centralized solution where the cloud or a more powerful device does the mapping is not viable since it requires knowledge (i.e., device's available resources, including stored data of user) of the entire network to produce a mapping.

IV. Resource management Technical Framework

Our technical framework is composed of two modules: (i) *the deployment policy module*, a novel decentralized task allocation technique that deploys tasks without prior network topology knowledge, and (ii) *the bidding policy module*, consisting of multiple bidding strategies used by edge devices to take local decisions regarding its available resources.

A. *Deployment policy module*

The deployment policy module is designed to map tasks at the edge of the network by advertising the IoT application model to the neighborhood and decide how to distribute them between participants such that the application meets its requirements. The procedure of finding a solution begin once a deployment request for an IoT application arrives. The dispatcher first prepares an advertisement message containing the application model and broadcasts it to participants. After all bids are collected, the dispatcher selects the winners based on application-wide objectives; a bid of a particular participant for certain tasks can be fully or partially satisfied.

We note that typically resource allocation solutions previously proposed to ensure that a set of objectives is satisfied for application tasks, use a centralized approach where the dispatcher node has "unlimited" (e.g., cloud) or limited but powerful computational resources [7], [8]. Due to the inherent problem complexity, such solutions use traditionally metaheuristic optimization algorithms. Those are a good choice when trying to find a near-optimal mapping. However, an implementation like this requires a longer computational time to yield a near-optimal solution that meets the application requirements. At the same time, if the simulation time is lower, we cannot guarantee that the obtained solution satisfies the application constraints.

In contrast to heuristic, ad-hoc solutions, we use a technique with formal foundations offering guarantees that a found solution always satisfies the application requirements. Our proposed technical framework utilizes satisfiability modulo theories (SMT) [9], a generalization of the boolean satisfiability (SAT) problem. Using SMT, we ensure that if a satisfiable solution exists, always satisfies the given constraints of latency and privacy. Hence, to solve our task allocation problem, we have created a SMT formula composed of five different parts: i.e., *tasks facts*, *domain facts*, *latency facts*, *bid constraints*, and *constraint formulation*.

In the *task facts* section, we define the rules required to place correctly a task on an edge node. First of all, we ensure that a task $\mathsf{t_i}$ can be placed on a specific node only if a bid for $\mathsf{t_i}$ has been sent by that node to the dispatcher. Moreover, since multiple participants can bid for the same task, the dispatcher must ensure that only one bid is considered when a solution is created. As a result, we avoid the case when in the placement solution a task is mapped on multiple edge nodes. As an example, consider that we want to deploy the IoT application, presented in Fig. 1 on two participant nodes $\mathsf{E_1}$ and $\mathsf{E_2}$. Let us assume that edge node $\mathsf{E_1}$ sends a bid for tasks $\mathsf{t_1}$, $\mathsf{t_2}$ and $\mathsf{t_4}$, while $\mathsf{E_2}$ sends a bid for $\mathsf{t_1}$, $\mathsf{t_3}$ and $\mathsf{t_5}$. In this case, we ensure that $\mathsf{t_2}$ and $\mathsf{t_4}$ can be deployed only on $\mathsf{E_1}$ and $\mathsf{t_3}$ and $\mathsf{t_5}$ can be deployed only on $\mathsf{E_2}$.

The next two parts, *latency and domain facts*, constraint further a possible mapping, by encoding communication latency between tasks that have dependancies. In *latency facts*, we initialize a set of symbols with the communication latency between edge nodes in that area. This is an important step in the creation of the final formulae encoding, since all required knowledge must be available prior to finding a solution. Considering the example above, symbol $\mathsf{s_{E1E2}}$ represents the latency $\mathsf{l_{E1E2}}$ between $\mathsf{E_1}$ and $\mathsf{E_2}$. Once the latency between nodes is known, we create the *domain facts* encoding formulae, which captures latency between two tasks. Thus, if a task $\mathsf{t_1}$ is mapped to a specific node $\mathsf{E_1}$ and $\mathsf{t_2}$ is mapped to $\mathsf{E_2}$, it implies that the latency between the two tasks $\mathsf{l_{t1t2}}$ is equal to $\mathsf{s_{E1E2}}$.

Next we create a set of rules with the purpose of aiding the dispatcher in choosing a valid solution, that does not exceed the available resources of the participants, based only on the received bid. As a result, we have created a new encoding, i.e., *bid constraints*, that limits the winner tasks to the boundary of one offer. Let us consider the following scenario; a participant $\mathsf{E_1}$ decides that it can process some tasks of the motivation example application and creates a bid containing three offers, B = $\mathsf{o_1}$, $\mathsf{o_2}$, $\mathsf{o_3}$, where each offer is described as follow: $\mathsf{o_1}$ =[$\mathsf{t_1}$, $\mathsf{t_3}$], $\mathsf{o_2}$ =[$\mathsf{t_1}$, $\mathsf{t_2}$], and $\mathsf{o_3}$ =[$\mathsf{t_5}$].

Based on the set of rules described above, the dispatcher can select only tasks from one offer to be mapped on $\mathtt{E}_1$. Now let's assume that the dispatcher choose $\mathtt{t}_1$ as the first task to be mapped on $\mathtt{E}_1$. As a result, $\mathtt{o}_3$ is blocked since it does not contain $\mathtt{t}_1$, remaining to choose from the available tasks from the other two offers since they both contain $\mathtt{t}_1$.

Finally, the last encoding section of our SMT formula ensures that the solution satisfies the latency constraint. The *constraint formulation* contains the rules which the found solution must satisfy. Such constraints only account for the latency defined in the e2e delay information, which can be the network latency or some other latency that the developer specified for a specific application.

B. *Bidding policy module*

The bidding policy module has the purpose of generating an individual bid for the advertised tasks, enforcing privacy requirements by construction. This occurs because only bids satisfying privacy requirements are constructed by nodes that have privacy preferences. Generally, each bid sent to the dispatcher is composed of multiple offers generated based on the available strategies. Every participant has a total of four strategies with the purpose of providing wider coverage of the tasks model, by ensuring each task receives at least one bid and the system can ensure solutions for varying task graph types. Each of the four strategies has a different role in the overall bid. Therefore, we group the strategies conceptually based on their role. We abstain from formally defining them since they are based on well known algorithms from dynamic programming and graph theory.

1) The first group, composed of one strategy based on the *0-1 knapsack problem* and the other on a random selection of tasks, has the objective of maximizing the available resources of a participant at the cost of not being possible to win the entire bid since it will most likely not meet the application objectives.
2) The second group is focused on generating offers that have a higher chance of being fully satisfied since bids contain tasks dependent on each other. The *strongly connected components strategy* creates an offer selecting components in the task graph that utilize the maximum available resources. In contrast, the *fan-out strategy* aims to find the tasks with the biggest fan-out that can be locally mapped, and maximizes resource utilization by adding dependent tasks.

V. Evaluation

To provide concrete support for our framework, we realized a prototypical tool based on the Z3 SMT solver [10]. Thereupon, we evaluate our approach discussing its performance over an IoT application model by follow a quantitative approach. Finally, we conclude with a discussion of the obtained results.

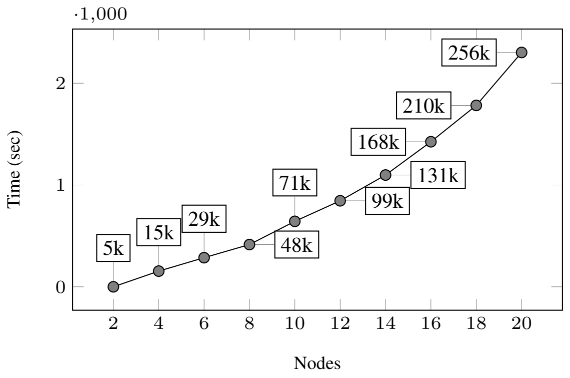

Figure 2. Mapping time over number of nodes and SMT formula size.

A. *Performance: Experiments Setup and Results*

We measure performance by the execution time required to obtain an allocation for tasks and its ability to find a satisfiable solution at the edge by the dispatcher node, as this is the higly computationally-intensive operation in our technique. For our experiment setup, we map an IoT application to an edge computing architecture where we incrementally increase the size of the participant devices. We randomly allocate to every node a set of available resources, chosen in the range of 10 to 20 units each. For the IoT application in our experiments, we utilize the *montage graph*, a real-world workflow [11] represented as a DAG. The application is composed of 24 tasks and 50 edges, for which we synthesize resource constraints in the range of 1 to 10. Regarding other constraints of this application, i.e., SLA and required data, we set the SLA to a large value to ensure that found solutions are not discarded due to resource randomization. We ignore data requirements of tasks to maximize coverage of each task from bids, thus each node is able to bid for all tasks with no restrictions.

We measure performance of our framework by deploying the dispatcher on a machine with a single-core Intel i5 2.3GHz processor. We perform 100 tests for edge architectures of different sizes. We measure the time it takes to calculate the mapping over different number of participating nodes. Our results are shown in Fig. 2.

B. *Discussion*

With the aid of our experiments, we have demonstrated that our framework is efficient in finding satisfiable solutions in a decentralized resource management scenario. From the results presented in Fig. 2, the impact of the number of participants nodes and of the SMT formula size on the execution time is illustrated. In this case, we can observe that the three parameters are closely bound. The formula size grows since the *bid constraints*, *latency facts*, and *domain facts* increase with the number of nodes and tasks. Hence, the time to find a solution rises as well.

Even though the evaluation is performed on a laptop, we were able to correctly evaluate the performance of our framework. We note the high computational demands of our technique, but point out that for relevant problem sizes, it is highly feasible and it exhibits guarantees – if a mapping exists, it is found.As a result, because the framework becomes very computational demanding after 20 edge nodes, a suitable realistic size for the neighborhood is between 5 to 15 nodes. In this range, high success rate in a relatively small execution time (less than 1 second) is obtained.

VI. Related work

Recently, research has focused on proposing resource allocation solutions to utilize the newly available resources found closer to end users and devices. Mobile Edge Computing (MEC) is one area where researchers have used auction-based mechanisms to offload parts of an IoT application. In [12], authors propose an auction-based mechanism to perform resource allocation to MECs and compute the afferent price for each resource. Similarly, authors in [13] present a double auction-based service to match the requests of the users to the available resources of a MEC. Our auctioning framework differs by creating bids with the purpose of sharing resources, while the dispatcher only decides where to deploy the tasks.

In the context of using the available resources found on multiple edge devices, authors in [7] propose a collaborative approach where a pre-partitioned application is distributed among edge nodes such that communication latency is minimized, while [14] describes a competitive-cooperative game-theoretic resource allocation framework to deploy latency-sensitive application at the edge, ensuring cooperation between nodes by offering incentives based on their work. Compared to the two solutions presented above, our technical framework differs from two perspectives; (i) we guarantee that if there is a solution possible, it is always found in a dependable manner, and (ii) we maintain data privacy requirements by sending tasks to process privacy-sensitive data locally at its data source.

VII. Conclusion

In this paper, we proposed a novel decentralized resource management technique and accompanying technical framework for deployment of latency-sensitive IoT applications at the edge of the network. Our technique is inspired from the functionality of an auction house; our results show that our decentralized resource management technical framework can efficiently utilize the available resources at the edge and provide guarantees – if there exists a solution that satisfies application latency and privacy objectives, it will be found, as it amounts to a satisfiability problem. Regarding future work, we plan to optimize our encoding with the aim of reducing execution time, while increasing further the accepted number of participant nodes.

Acknowledgment

Partially funded by the EU's H2020 research and innovation programme under Marie Skłodowska-Curie grant agreement No. 764785– FORA; the TU Wien Research Cluster SmartCT; the Austrian Federal Ministry of Science in the CPS/IoT project.

References

[1] S. Wang *et al.*, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, 2017.

[2] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. June, pp. 30–39, Jan 2017.

[3] K. Toczé and S. Nadjm-Tehrani, "A taxonomy for management and optimization of multiple resources in edge computing," *CoRR*.

[4] V. D. Maio and I. Brandic, "First hop mobile offloading of dag computations," in *18th IEEE/ACM Intl. Symposium on Cluster, Cloud and Grid Computing*, 2018.

[5] J. Ren *et al.*, "Serving at the edge: A scalable iot architecture based on transparent computing," *IEEE Network*, 2017.

[6] M. Deng, H. Tian, and B. Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *IEEE Intl. Conf. on Communications Workshops*, 2016.

[7] M. M. Shurman and M. K. Aljarah, "Collaborative execution of distributed mobile and iot applications running at the edge," in *Intl. Conf. on Electrical and Computing Technologies and Applications*, 2017.

[8] M. Aazam and E. N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *IEEE Intl. Conf. on Pervasive Computing and Communication Workshops*, 2015.

[9] C. Barrett and C. Tinelli, *Satisfiability Modulo Theories*. Springer International Publishing, 2018.

[10] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Intl. conf. on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[11] L. F. Bittencourt, R. Sakellariou, and E. R. M. Madeira, "Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm," in *18th Euromicro Conf. on Parallel, Distributed and Network-based Processing*, 2010.

[12] T. Bahreini, H. Badri, and D. Grosu, "An envy-free auction mechanism for resource allocation in edge computing systems," in *IEEE/ACM Symposium on Edge Computing*, 2018.

[13] H. Zhang, F. Guo, H. Ji, and C. Zhu, "Combinational auction-based service provider selection in mobile edge computing networks," *IEEE Access*, 2017.

[14] D. Zhang *et al.*, "Cooperative-competitive task allocation in edge computing for delay-sensitive social sensing," in *IEEE/ACM Symposium on Edge Computing*, 2018.