Latency-aware decentralized resource management for IoT applications

Cosmin Avasalcai
Distributed Systems Group
TU Wien, Vienna, Austria
c.avasalcai@dsg.tuwien.ac.at

Schahram Dustdar
Distributed Systems Group
TU Wien, Vienna, Austria
dustdar@dsg.tuwien.ac.at

ABSTRACT
With the increased success of the Internet of Things (IoT), the cloud-based solutions are no longer sufficient to meet the stringent requirements of IoT applications. Besides requiring fast response time, increased security and privacy, they lack computational resources at the edge of the network. Such computational resources distributed closer to the edge represent the fundamental infrastructure of edge and fog paradigms. However, without advanced resource management techniques, the advantages of the paradigms are lost. In this paper, we present a novel distributed resource allocation algorithm with the purpose of enabling seamless integration and deployment of different applications in an IoT infrastructure. The algorithm decides: the (i) mapping of IoT application at the edge of the network, (ii) dynamic migration of parts of the application, such that Service Level Agreement (SLA) is satisfied. Furthermore, we analyze and discuss our approach and the potential to minimize the latency of different IoT applications.

ACM Classification Keywords
C.2.1 Network Architecture and Design: Distributed networks. Network topology; C.2.4 Distributed Systems: Distributed applications; D.4.7 Organization and Design: Distribution systems;

Author Keywords
Edge Computing; Internet of Things; Resource Management; Fog Computing.

INTRODUCTION
The continuum increase of connected devices introduced by the IoT and the need for real-time applications with fast response times, higher privacy, and security, have pushed the horizon of new computing paradigms, fog, and edge computing. Fog computing paradigm has been coined by Flavio Bonomi in 2012 and extends the cloud services capabilities to use computing resources near IoT sensors, decreasing network congestion and minimizing the latency of real-time applications [1]. In contrast, edge computing pushes the frontier of deploying heterogeneous computational resources near the edge of the network, even closer to the source of data [6].

However, with no support from novel resource management techniques, the paradigms alone cannot ensure a good deployment of applications at the edge. These techniques come as a solution to the constraints imposed by the IoT devices (i.e., heterogeneous devices, with limited computational resources and energy). Moreover, a higher degree of uncertainty in the system is introduced by the mobile devices (e.g., smartphones, tablets, cars, etc.) which can enter and leave freely the network without any prior announcement. Hence, to fully utilize the available resources found at the edge of the network, to reduce communication bandwidth, improve security and enable real-time IoT applications, new resource management algorithms must be created. Some use cases that benefit from deploying IoT applications at the edge are, remote-monitoring in the healthcare area, controlling real-time systems in smart factories and deploying applications in smart cities. In our proposal, we focus on developing a novel decentralized algorithm to tackle resource allocation from three different perspectives, (i) a new deployment strategy that offers a personalized mapping for each IoT application independent of environment or location, (ii) a self-adapt algorithm at runtime to improve the reliability of IoT application by migrating tasks from faulty nodes and (iii) a ranking system to help the algorithm decide what tasks should be migrated and when.

RELATED WORK
In the last years, with the advent of edge and fog computing introduced the need for new resource management techniques at the edge of the network. Therefore, more and more researchers proposed solutions to tackle some of the challenges introduced by this new IoT infrastructure. Jain et al. [2] proposes a mapping algorithm that divides the IoT application into multiple tasks annotated with location information based on which is deployed at the edge. Another approach which uses the edge resources is suggested in [7], where a pre-partitioned application is distributed among edge nodes such that the communication with the cloud is reduced and the latency is minimized.

The authors in [3] introduce a cooperative fog platform that tries to ensure an easy collaboration between multiple static and mobile fog devices by using a distributed communication model. Moreover, to improve the service efficiency of IoT applications, an allocation algorithm is used to select the host based on the characteristics of the system. A similar approach

IOT '18, Oct 15–18, 2018, Santa Barbara, CA, USA
©2018 Copyright is held by the owner/author(s).
ACM ISBN 978-1-4503-6564-2/18/10.
DOI: https://doi.org/10.1145/3277593.3277637

is presented in [8], where an optimization service placement algorithm is developed to share fog resources. However, the algorithm will always try to map an application to the deployment node or to a neighbor device. In case of failure, the application is mapped to the cloud.

In [4], the authors propose an algorithm to dynamically migrate virtual machines (VMs) and find the best communication paths based on predictions of user movements. A similar solution to service migration is presented in [5], where the authors propose an edge-enabled publish-subscribe middleware to continuously monitor the QoS and transparently migrate clients to another host in close proximity.

With respect to the aforementioned research papers, most of the researchers' work is emphasizing the placement of edge devices. In contrast, our proposal provides a more comprehensive resource management solution that is independent of IoT applications and environment. By combining the initial placement of the application at runtime, with dynamic migration and neighborhood selection, the edge devices will become more intelligent.

PROPOSED SOLUTION

In this paper, we propose a decentralized algorithm designed to offer the developers the possibility of deploying IoT applications at the edge of the network without knowing the characteristics of the edge devices in that particular location. This approach improves the security of the IoT application and nodes since even the developer does not know on which nodes the application is mapped. Moreover, the algorithm is in charge of monitoring the overall performance of the application, improving its reliability by dynamically adapt to changes in the network.

Due to the limited resources of edge devices, we assume that the application is divided into multiple tasks (i.e., a set of instructions that perform a specific computation) by the developer prior to deployment. Hence, we model an IoT application as a Directed Acyclic Graph (DAG) where vertices represent different tasks and edges describe the dependencies between them (see Figure 1).

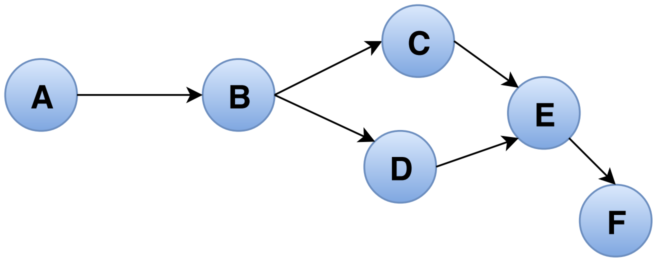

Figure 1. IoT application model

We consider that the IoT architecture is represented by a peer-to-peer communication between each IoT device and the cloud (see Figure 2). Compared to the common representation of such an IoT architecture (i.e., a four-layer pyramid starting with the sensor layer and ending with the cloud), our model is a flatter design that offers the possibility of sharing resources vertically and horizontally between devices.

The behavior of our algorithm is inspired by the organization of a private auction house. Considering this, we have divided our algorithm into three different modules: neighborhood latency monitoring, bidding policy, and placement policy.

Neighborhood latency monitoring

This module creates for each individual IoT application that has to be deployed, at runtime a list of nodes chosen from the neighborhood (i.e., all the nodes that have a direct connection with the dispatcher) and their related latency. A dispatcher node represents the node that received a request to deploy the IoT application in that location. To choose the nodes, the module relies on an algorithm for finding the latency smaller or equal to a computed threshold (i.e., specific to each application, since it represents a percentage of the total E2E delay). Once a request arrives from the placement policy module, the list of nodes is generated and sent back to the placement module.

Furthermore, the module has two more tasks to perform. First, it monitors and stores the latency of each IoT device found in the neighborhood. Second, it notifies the placement module if the latency to a winner node (i.e., a node that received a task to compute from the deployed application) has increased or if it is unresponsive. In both cases, a migration of that affected tasks is performed.

An example of the individual lists created for two different IoT applications is presented in Figure 2. As you can see, in this case, the IoT application deployed from the yellow edge device has a smaller list of devices to borrow resources since its requirements are more strict. In contrast, the application deployed from the purple edge node can deploy to multiple nodes, because of its requirements.

Bidding policy

The bidding module generates offers and sends them to the dispatcher. Based on these offers, the dispatcher deploys the application such that the SLA is satisfied.

The algorithm starts when a broadcast message arrives from the dispatcher node. Based on the available resources and task dependencies, the module selects a number of tasks that can compute and creates the offer. However, the selection of tasks has an important constraint, i.e., a node can select only tasks that share a dependency. Once the number of tasks is selected, the module creates a latency table containing the communication latency from the current node to the list of nodes (i.e., the devices selected by the monitoring latency module in the dispatcher when the IoT application deployment request arrived) received from the dispatcher. To obtain this table, the algorithm performs an intersection between the received list and its neighborhood latency table. This information is very important to the deployment. Finally, after the latency table is computed and the tasks are selected, an offer is sent to the dispatcher.

Placement policy

The module proposes an optimization reactive placement with the purpose of distributing the IoT application and dynamically

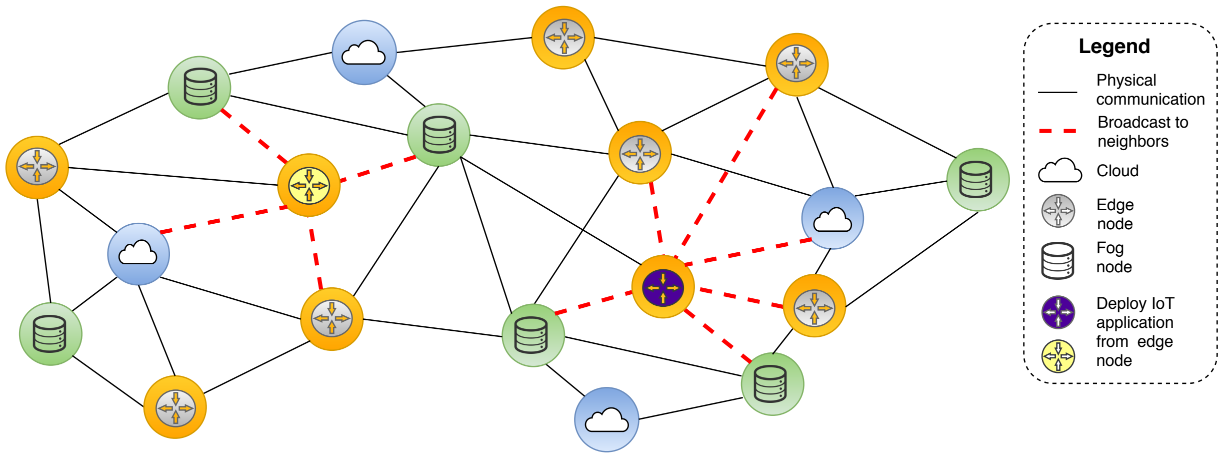

Figure 2. IoT architecture

adapt the mapping at runtime. Once a request for an application has arrived at a dispatcher, the algorithm distributes the tasks of an application such that the E2E delay is minimized.

The first step of the algorithm is to identify if there are any computational demanding tasks to be mapped to the cloud. Next, for the remaining tasks, the algorithm tries to map one or all tasks locally. Since processing tasks locally reduce significantly the overall E2E delay, it is mandatory to map at least one task on the dispatcher. In the case where no resources are available, the rank of the currently mapped tasks is used to decide which one to migrate. At the same time, the dispatcher of the chosen task is notified to find a new placement node.

Once the list of IoT devices is received from the neighborhood latency monitoring module, the algorithm creates and broadcast a message to each individual device. The message contains the remaining tasks and their dependencies, the SLA of the IoT application and the list of devices just received. However, before sending the message, a timer is started to limit the waiting time for offers. In this case, all offers arrived after this period of time will be discarded.

Next, after the time has expired and all offers are collected, the best deployment strategy is computed by choosing the offers that yield the smallest latency using Equation 1:

$$App_{latency} = T_{map} + B_{latency} \tag{1}$$

where T_{map} represents the time to find the best mapping for the IoT application and $B_{latency}$ is the sum of the communication latency between the winner nodes. It is important to state that T_{map} only impacts the E2E delay when the application is deployed. In any other case, the T_{map} is equal to 0.

Finally, a message is created containing the tasks docker container and the addresses of their dependency, providing a direct communication link with these nodes. Using docker containers to send the tasks over the network gives the best solution to ensure that all tasks will work properly independent on the node.

CHALLENGES DISCUSSION

In this section, we discuss the challenges faced when designing our decentralized algorithm. To unlock the true potential of the proposed algorithm and enable more computational resources at the edge of the network, we have to perform extensive evaluations of the entire system and for each individual modules.

Challenges in neighborhood latency monitoring

As described in Section 3 the purpose of this module is to create a specific list of nodes for each IoT application that is deployed from a dispatcher. This module has two main central pieces that are utmost important to the overall behavior. The first piece is represented by the value of the threshold. Since this determines the total number of devices that can share resources for a given IoT application, it has a big impact on the computational time of finding and how the deployment strategy looks. For example, if we choose a bigger percentage then a bigger number of devices will be accepted. Thus, it provides more resources to choose from with a penalty of computational time. In the other extreme, the placement is faster, but we could end up mapping everything to the cloud. The second core piece is the latency monitoring part, where we must find a solution to monitor the latency without introducing a communication overhead.

Challenges in bidding policy

This module generates the offer submitted by a node to the dispatcher. Also, it is the module that has a great impact on the overall mapping, since the offers reflect the devices which have free resources to share and the knowledge that the placement policy will use to determine the best mapping. However, there are two important aspects when developing this module that needs our attention, the ranking system and ensuring that we receive offers for all tasks.

The ranking system plays an important role in deciding what tasks to migrate. Thus, each accepted task should receive a rank based on the following characteristics: trustiness of the dispatcher, communication latency, resource utilization, etc. These rank ultimately shows how good is the placement of the task on that particular node.

The other aspect that we must discuss is how to ensure that a dispatcher receives offers for all the tasks, such that a solution can be created. A possible solution is to broadcast the offer to the other participant nodes before sending it. Hence, when a device receives a broadcast offer he can adjust his offer accordingly, making sure that there are offers for all tasks. However, this could impact the communication overhead.

Challenges in placement policy module

The placement module is in charge of selecting the best offers such that distribution strategy meet the requirements of the application. The first challenge is related to mobile devices and their resources. In order to truly use all available resource at the edge, these devices have to be considered. Besides the legal and security aspect of sharing resources from such devices, handling their unpredictability introduced into the system is utmost important. For example, one possible solution is to find an additional stationary device to map the task once the mobile device has disappeared from the system (either by shutting down or move to another location). However, this approach does not ensure that the stationary device has free resources when the mobile node is down. Thus, more tests and discussions have to be done to solve properly this problem.

Another important aspect is the period of time that the dispatcher waits for offers. As you can imagine, choosing this value has a great impact on how good the placement is. If the time is too low, then we could lose important offers. If it is too high, then we increase the computational time. However, this value could be individual to each IoT application and its related characteristics. In this case, how to make a difference and what characteristics to use to obtain this time?

Overall challenges of the system

For the overall system design, we want to introduce and discuss possible improvements and solution that could make more efficient the utilization of resources at the edge of the network. One such improvement consists of migrating the IoT application at other location where are needed. For example, some applications are used only for a limited period of time during a day, a month or a year. Thus, offering the possibility to free allocated resources when the application is not functional could push the importance of edge computing even more. Another important aspect that we must avoid is represented by migration loops. Such an event can occur when one or more tasks are migrated continuously.

Finally, finding what is the maximum concurrent number of deployments from different edge nodes located close to each other influences the efficiency of the proposed algorithm. Since this limit shows us the overhead that the message exchange is having on the network.

CONCLUSION

The most efficient approach to satisfy the demanding real-time requirements of IoT applications is to enable more computational resources at the edge of the network. However, introducing new paradigms to bridge the cloud and edge is not enough. A distributed resource management technique that can adapt to individual needs and environment of different applications is required. In this paper, we introduce a distributed load balancing algorithm that aims to satisfy the latency constraints of each application by collaborating with nearby devices to ensure minimum latency. We truly believe that applying the proposed algorithm in a real IoT infrastructure (i.e., smart city, smart factory, etc.) can provide a seamless integration of all applications.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA–Fog Computing for Robotics and Industrial Automation. This work also has been partially supported and funded by the Austrian Research Promotion Agency (FFG) via the "Austrian Competence Center for Digital Production" (CDP) under the contract number 854187.

REFERENCES

1. Flavio Bonomi et al. 2012. Fog computing and its role in the internet of things. *1st ACM Mobile Cloud Computing Workshop* (2012), 13–15.
2. R. Jain and S. Tata. 2017. Cloud to Edge: Distributed Deployment of Process-Aware IoT Applications. In *2017 IEEE International Conference on Edge Computing (EDGE)*. 182–189. DOI: http://dx.doi.org/10.1109/IEEE.EDGE.2017.32
3. A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis. 2017. A Cooperative Fog Approach for Effective Workload Balancing. *IEEE Cloud Computing* 4, 2 (March 2017), 36–45. DOI: http://dx.doi.org/10.1109/MCC.2017.25
4. J. Plachy, Z. Becvar, and E. C. Strinati. 2016. Dynamic resource allocation exploiting mobility prediction in mobile edge computing. In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 1–6. DOI: http://dx.doi.org/10.1109/PIMRC.2016.7794955
5. T. Rausch, S. Nastic, and S. Dustdar. 2018. EMMA: Distributed QoS-Aware MQTT Middleware for Edge Computing Applications. In *2018 IEEE International Conference on Cloud Engineering (IC2E)*. 191–197. DOI: http://dx.doi.org/10.1109/IC2E.2018.00043
6. W. Shi and S. Dustdar. 2016. The Promise of Edge Computing. *Computer* 49, 5 (May 2016), 78–81. DOI: http://dx.doi.org/10.1109/MC.2016.145
7. M. M. Shurman and M. K. Aljarah. 2017. Collaborative execution of distributed mobile and IoT applications running at the edge. In *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. 1–5. DOI: http://dx.doi.org/10.1109/ICECTA.2017.8252057
8. Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. 2017. Optimized IoT service placement in the fog. *Service Oriented Computing and Applications* 11, 4 (01 Dec 2017), 427–443. DOI: http://dx.doi.org/10.1007/s11761-017-0219-8