Fides: A Hidden Market Approach for Trusted Mobile Ambient Computing

Dimitris Chatzopoulos*, Pan Hui*, Di Huang*
* Department of Computer Science and Engineering, Hong Kong University of Science and Technology

***Abstract*—We propose Fides, a computation offloading mechanism for mobile devices based on trust between them.Fides provides functionalities to the application developers in terms of (i) application splitting and (ii) metric proposal, according to which the offloading decisions will be made. A collective intelligence mechanism is learning the parameters of the ambient environment and selects the proper nearby devices (helpers) to aid the device that is asking for help (aided) via a linear complexity online algorithm. Experiments in android mobile devices show how user's quality of experience can be improved in terms of battery consumption and delay minimisation.**

I. Introduction

The needs of modern mobile applications motivate research works in mobile cloud computing where parts of the applications are offloaded to more capable cloud servers [1], [2], [3]. However, the use of remote servers implies Internet access which can not be available or cheap and also can be energy harvesting [2]. Furthermore, network saturation in popular locations is reality and motivates the research activity in 5G technologies where one of the main goals is to solve the bandwidth sharing problem in crowded environments. Concluding, the connection between two smart devices in the same area using WiFi direct can be more energy efficient and reliable.

The necessity of Fides comes from the fact that the well-behaving participants (*helpers*) should be acknowledged and helped while free riders should be put aside. Each of its layers works transparently and autonomously following the principles of hidden market design [4]. Fides explores and connects with nearby devices (**CONNECT**), learns and estimates their trustworthiness (**LEARN**) and provides a low complexity algorithm for computation offloading (**DECIDE**). Users who wish to expose their computational capabilities to the mechanism have only to specify the amount of their computational resources, a lower bound on their smartphone utilization and an upper bound on their battery level via a simple interface.

II. Mobile Application

Most mobile applications are built via an object oriented programming language, which means that they are based on a set of classes. A subset of them can be executed by another device (*offloadable*). We expect from the developers who use Fides mechanism to define all the possible points

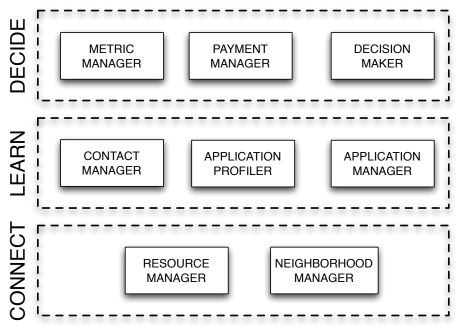


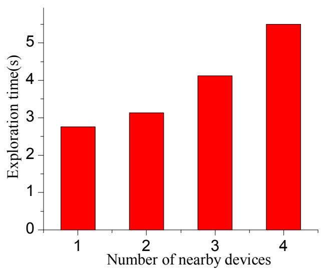
Fig. 1: The three-layer architecture of Fides.

in the application execution where the application can be split. In Java programming this can be done by putting the flag *@splitable* at the definition of a method that can be executed to another device and the flag *@split* at the call of the splitable methods that she wants to execute to another device. Some representative applications that could benefitted by Fides are video compression, feature extraction in image processing (e.g face recognition and scene understanding in surveillance systems) or mobile augmented reality applications like Microsoft HoloLens.

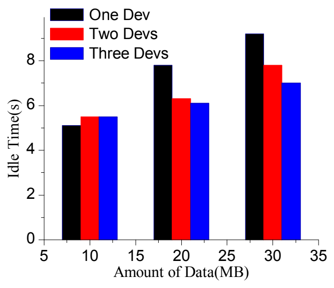
III. Online Algorithm

A. Credit Managment and Neighbors' Reputation

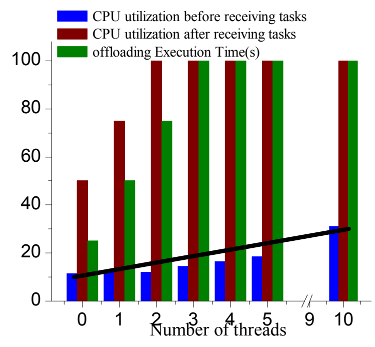
In order to use the Fides mechanism, one user has to earn some credits first. Every device uses the application profiler (AP) to calculate the extra burden of an additional task. Given the current state and the new state AP estimates the cost of serving the request for help. The extra burden depends on the recourses as well as the network overhead to run the task. After estimating the computational cost, any nearby user bids to the interested for offloading user and given the bids and the reputation of each user she selects the proper ones using the algorithm that presented below. Regarding the reputation calculation contact manager encapsulates a collective intelligence scheme where each user in the neighborhood, who participated in a code offloading that has just finished, broadcasts her updated opinion for any user with whom she interacted. After the reception of information by a trusted friend, any user updates her opinion as well.



(a)



(b)



(c)

Fig. 2: The time needed to explore the neighborhood and the idle time while waiting for the results are increasing with the number of nearby devices and the result size. Helper's delay response depends on her current utilization and in the number of running threads.

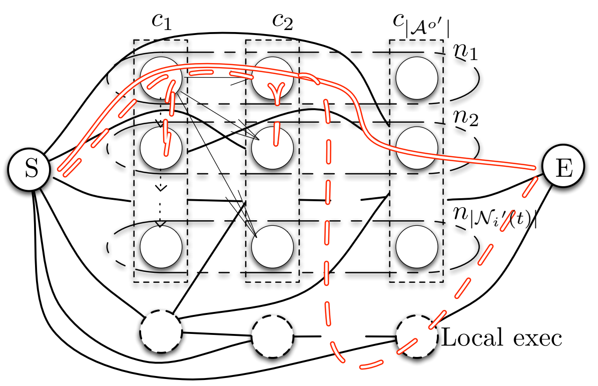


Fig. 3: State Diagram of the proposed online algorithm.

B. Algorithm

We propose an online algorithm with linear complexity to the number of offloadable classes and to the possible helpers and runs in the **DECIDE** layer of Fides. The inputs to the algorithm are a credit budget and a trust score and the output is a set of helpers. Its most heavy part is a preprocessing part which sorts the neighbors list, which detected by the **CONNECT** layer, and the classes list and can be executed proactively by the **LEARN** layer of Fides. Figure 3 depicts the algorithm's functionalities and state space where for visualization reasons only a representative subset of the edges is shown. Each edge between two nodes in the state space represents the metric cost of executing one class at one helper given that another class will be executed at another helper. We expect links between the same helpers to have lower metric cost because the execution of two classes in the same device is lower that the execution of the same classes in two different devices because of the lower networking cost. Of course this also affects the credit cost. The two red and bold lines indicate two possible solutions of different scenarios. The continuous line indicate a solution in which all the offloadable classes are executed remotely while in the dashed solution the budget is not enough for all the classes and only the most critical are offloaded.

IV. Implementation & Evaluation

We implement the basic functionalities of Fides using Android APIs and prototyped it on Google Galaxy Nexus, Samsung Galaxy SII, and Motorola Moto G. To perform the task offloading, we use, WiFi direct to establish the connection between the devices.Fig. 2a shows that the time needed to explore the neighborhood and find the nearby devices is increasing in the number of the devices. Fig. 2b depicts the idle time of aided device while waiting for the results of the offloaded classes as a function of the data needed by the class to execute properly. Each color of Fig. 2b represents a different number of helpers. The exploration time is increasing in the number of nearby devices while the idle time is decreasing because the application is divided into smaller parts. Fig. 2c shows how the current utilization and the number of the threads in the helper affects the response of the helper in terms of delay. The x-axis is the number of the threads in the helper. To produce this plot we offloaded a BubbleSort instance of 20000 integers in a quad core Moto G.

References

[1] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *INFOCOM, 2012 Proceedings IEEE*, 2012, pp. 945–953.

[2] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10, 2010, pp. 49–62.

[3] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11, 2011, pp. 301–314.

[4] S. Seuken, D. C. Parkes, E. Horvitz, K. Jain, M. Czerwinski, and D. Tan, "Market user interface design," in *Proceedings of the 13th ACM Conference on Electronic Commerce*, ser. EC '12, 2012, pp. 898–915.