Mobility-Aware Application Scheduling in Fog Computing

Luiz F. Bittencourt, Member, IEEE
Javier Diaz-Montes, Senior Member, IEEE
Rajkumar Buyya, Fellow, IEEE
Omer F. Rana, Senior Member, IEEE
Manish Parashar, Fellow, IEEE

Fog computing provides a distributed infrastructure at the edge of the network, resulting in low-latency access and faster response to application requests. With this new level of computing capacity, new forms of resource allocation and management can be developed to take advantage of the fog infrastructure.

Computing requirements of mobile users continue to increase, as computing and communication capabilities of smart and wearable devices and in-vehicle systems continue to improve. Many applications rely on remote resources to off-load and complete tasks, primarily through the use of a large-scale computing facility hosted within a data center. Such cloud systems are also able to support applications by storing data

 2325-6095/17/$33.00

and processing tasks offloaded by mobile or fixed devices.

With increasing focus on Internet-of-Things (IoT), countless devices scattered and connected to the Internet, producing and consuming data requires scalable resource management at unprecedented levels.[1] The data dynamism and heterogeneity resulting from this expected explosive expansion of connected devices, commonly referred in a broad sense as *Big Data*, also requires new processing models and infrastructures to support its main dimensions: data volume, velocity, and variety. One key aspect of this new era is that both data consumption and production are heavily distributed and at the edges of the network (i.e. closer to or at end-user devices). While the centralized data center model of cloud computing can cope with many types of applications and large amounts of data, its infrastructure and network connection to the edge are not designed to handle this *Big Data* phenomenon. In this context, computing and data management models that support computing capacity at the edges of the network are now a focus of significant research. Mobile clouds, vehicular networks, and fog computing are examples of new distributed computing models that leverage edge capacity closer to data production.[2]

With data also produced at the edge, both data generation and consumption can occur at many different places and times. In this context, different applications can have different requirements, especially in terms of response time. Currently, applications often rely on the cloud to have data and processing support, which may not be suitable for lower latency requirements. Moreover, execution of applications in cloud data centers does not take user mobility into consideration, and data or processing of an application can occur at a geographically distant data center. On the other hand, in a distributed computing scenario at the edges of the network, data distribution and processing can be maintained closer to the user, reducing network traffic to data centers and improving application response times as a result of lower network latency or delay.

This paper discusses the problem of resource allocation considering the hierarchical infrastructure composed of edge capacity and cloud data centers, analyzing application classes along with different scheduling policies. To address this challenge, we introduce a number of scheduling approaches that consider user mobility and edge computing capacity, in the context of a Fog Computing infrastructure.[1] We discuss the benefits of combining the application classes with scheduling policies in scenarios that illustrate these scheduling approaches, especially in the context of user mobility.

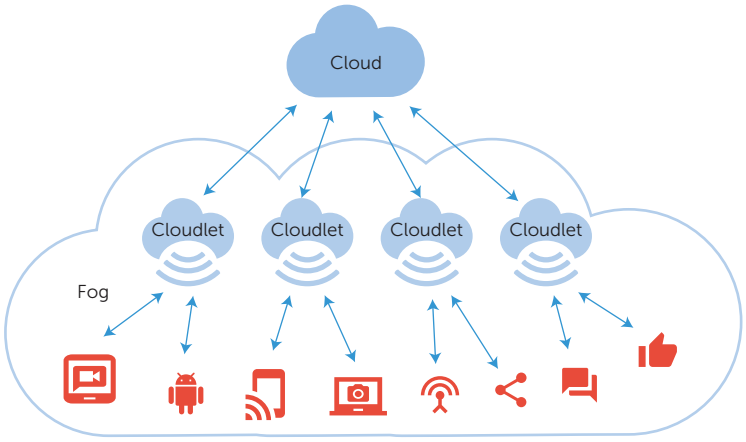

FIGURE 1. Fog computing: cloud, cloudlets and edge devices/applications ecosystem.

The next section presents the fog computing model considered in this paper and then related work is discussed. Next, we introduce the application model and two example applications, which are used in the evaluation further in the paper. After that, we present different allocation policies and simulation results. Then, we discuss resource allocation challenges for fog computing.

Fog Computing Model

User applications that access the public cloud do so through an access point that allows data exchange through the core network to reach the cloud data center. With the introduction of computing capacity at the edge of the network, these access points can be extended to also provide computing and storage services: the *cloudlets*. Figure 1 illustrates the cloudlets concept within the hierarchical infrastructure of the fog. This fog computing architecture presents a hierarchical, bi-directional computing infrastructure: edge devices communicate with cloudlets and cloudlets communicate with clouds. Cloudlets can also

communicate with each other to perform data and process management in order to support application requirements, and to exchange fog control/management data (such as user device and application state).

In fog computing, processing and storage capacity is one hop away from the data production/consumption, which can benefit different types of applications

- Applications with low latency requirements, such as pedestrian and traffic security, surveillance, applications for vision, hearing, or mobility-impaired users, online gaming, augmented reality, and tactile computing can benefit from lower latencies because of a single hop connection to a cloudlet.
- Applications that currently rely on the cloud can also benefit from lower delays and response times when adopting a fog-based deployment if their data and processing is carried out by a nearby cloudlet. This can also reduce data traffic to the cloud.
- Raw data collected by many devices often does not need to be transferred to the cloud for long-term storage: data can be processed, filtered, or aggregated to extract knowledge and produce reduced data sets, which in turn are to be stored; or it can be processed and utilized right-away to other edge devices in the so-called sensor/actuator loop. In both cases, the fog computing paradigm can reduce network traffic from the edge to data centers.

Cloudlets can provide reduced latencies and help in avoiding/reducing traffic congestion in the network core. However, this comes at a price: more complex and sophisticated resource management and scheduling mechanisms are needed. This raises new challenges to be overcome, e.g., dynamically deciding what, when, and where (device/fog/cloud) to carry out processing of requests to meet their quality of service requirements. Furthermore, with smart and wearable devices, such mechanisms *must* incorporate mobility of data sources and sinks in the fog. Traditional resource management and scheduling models for distributed systems do not consider mobility and timeliness of data production and consumption in the resource management and allocation process. Fog computing scheduling must bring users location to the resource allocation policies to uphold the benefits of fog computing proximity to the user.

Related Work

Resource management and scheduling in fog computing is a new topic that combines aspects from sensor networks, cloud computing, mobile computing, and pervasive computing fields. In fog computing, sensors and other devices pervasively present at the edge of the fog generate data and consume data that have to be processed using the cloudlets and the clouds. Each of those fields has a plethora of literature, well documented by researchers.[3,4,5]

Fog computing has been discussed as a platform to provide support for Internet of Things (IoT) and analytics. Bonomi et al. discuss many aspects of a fog infrastructure, including the interplay between fog and cloud systems. The authors claim that fog computing can better address applications and services that do not fit well in the cloud, such as low-latency or geo-distributed applications. In this paper, we advance this discussion by introducing strategies that can be adopted in the presence of different application classes.

A programming model has been proposed to support fog computing applications, which includes event handlers and an application programming interface (API).[6] Programming models are complementary to the presented work, as they rely in application scheduling strategies to decide where to allocate the functions created by users.

GigaSight is a virtual machine-based cloudlet infrastructure used to support video storage analytics at the edge.[7] One motivation for the proposal is to avoid overwhelming metropolitan networks with a large amount of video streams sent to cloud providers. GigaSight applications can share the cloudlets with other types of applications, where a proper resource allocation can improve performance.

Jennings and Stadler[3] discuss resource management objectives and challenges in cloud computing. Many aspects are covered by them, including resource management functions and network-aware resource allocation, which are also relevant to fog computing. Edge devices in the fog introduce particularities that must be considered, among which application requirements and mobility are discussed in the remainder of this paper.

Multi-clouds are platforms that aggregate computing resources from different cloud providers.[8] While multi-clouds can help in decreasing latencies to the final users, they are not able to provide really low latencies as the ones advocated by fog computing with its 1-hop away cloudlet infrastructure. Thus, the hierarchical combination of fog and clouds (stand alone or federated) still offers advantages in terms of latency and network traffic.

The European Telecommunications Standards Institute (ETSI) has launched the Mobile-Edge Computing (MEC) initiative to create standards for

mobile edge computing platforms, having proposed a blueprint[9] and also documents presenting technical requirements, terminology, and service scenarios. A first specification is expected to be delivered by the end of 2016. In terms of resource allocation and scheduling, offloading decision in mobile devices is important.[5] Kosta et al. present ThinkAir, a framework to offload mobile applications to the cloud.[10] ThinkAir is able to parallelize execution using virtual machines, achieving reduced execution times and energy consumption in the mobile device. In a fog computing scenario, the offloaded modules should be allocated according to the application requirements, as we discuss further in this paper.

Applications in Fog Computing

The fog architecture is hierarchical, where processing and storage location decision is subject to application constraints and user geo-location. While the former can be specified in different ways, as in the form of quality of service (QoS) constraints, the latter depends on human (or other autonomous system) behavior. Ultimately, user behavior determines the time and position of a computing device, which along with QoS constraints can be used to create application classes that are relevant for resource management and scheduling in a fog computing environment. By acknowledging different application classes, one could employ different scheduling policies, algorithms, or mechanisms to deal with each class.

Application model

To illustrate how resource management in fog computing can benefit applications by considering geo-location and different application classes, we identify two types of apps: near real-time and delay-tolerant. For the former, we describe the electroencephalography (EEG) tractor beam game; for the latter, a video surveillance/object tracking application.

In the EEG tractor beam game (EEGTBG), players try to gather items by concentrating on them. A player that has a better concentration on an item can attract it towards him/herself. Fast processing and low response times achieved by edge computing devices can give players a true online, real-time experience.

The EEGTBG application has 5 modules EEG sensor, display, client, concentration calculator, and coordinator. The EEG headset senses user concentration and streams raw data to the client module. The client module filters/forwards consistent data to the concentration calculator module, which computes the concentration level of the user. The concentration level is sent to the client module, which updates the game display to the player. The coordinator module gathers and distributes measured concentration among players. The EEG sensor, display, and client modules are placed in the mobile device (e.g., smartphone). The concentration calculator and the coordinator modules can be placed in the cloudlets or at a cloud data center.

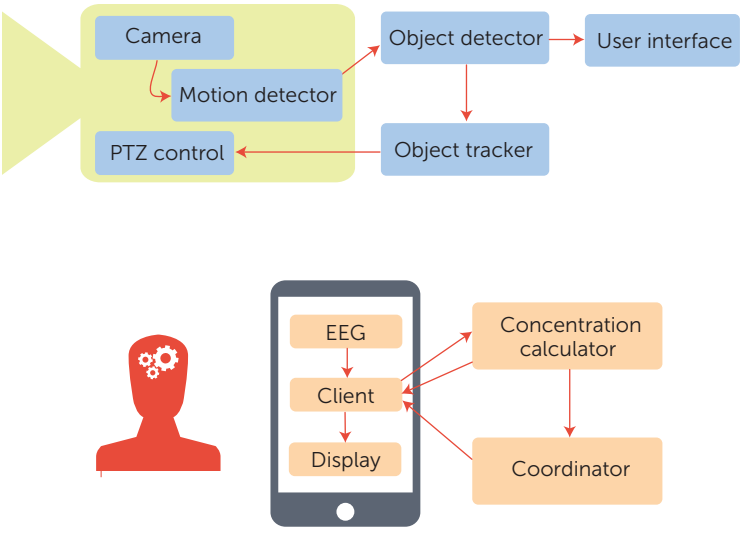

FIGURE 2. Example applications and their modules.

The video surveillance/object tracking application (VSOT) relies on a set of distributed intelligent cameras that are able to track movement, having 6 modules (see Figure 2): camera, motion detector, object detector, object tracker, user interface, and *pan, tilt, and zoom* (PTZ) control. The camera streams video to the motion detector module, which filters the incoming stream and forwards video in which motion was detected to the object detector module. The object detector module identifies the moving objects, sending object identification and position information to the object tracker, which in turn computes the desirable PTZ and sends the command to the PTZ control module. We consider that the motion detector and the PTZ control modules are always placed in the camera, while the user interface is always in the cloud. The object detector and object tracker are the two modules that should be placed by decision making policies in a cloudlet or at a cloud data center.[11]

Both applications – EEGTBG and VSOT – can be set up in a fog infrastructure to take advantage of lower latency due to the use of cloudlets. VSOT is able to work satisfactorily under data center-distance latencies (> 100 milliseconds). On the other hand, higher delays in EEGTBG can impact the players real-time perception, making the game unreal and impairing its playability. We consider that VSOT

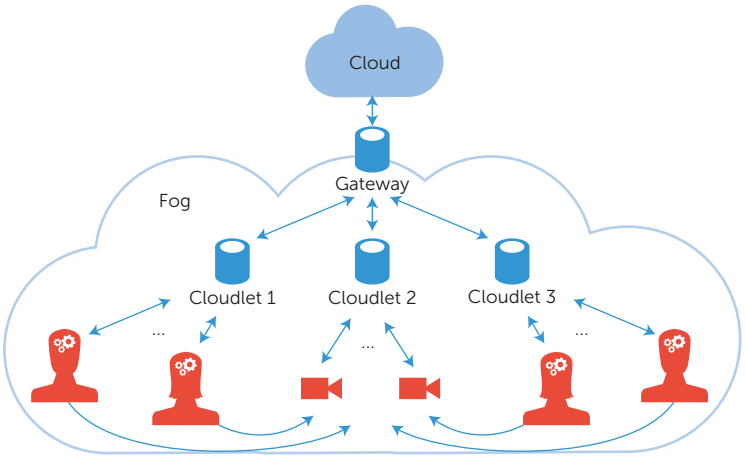

FIGURE 3. Mobility scenario: mobile concentration game users electroencephalography tractor beam game (EEGTBG) move and compete for the same cloudlet resources with existing surveillance (VSOT) application.

and EEGTBG belong to two different classes of applications (delay-sensitive and delay-tolerant) that can benefit from a fog computing infrastructure.

Mobility scenario

With the rapid use of mobile (smart) devices, fog computing infrastructure must be able to accommodate variable demands in the cloudlets, while relying on elasticity in cloud computing systems to offload processing and/or storage when necessary. This scenario is illustrated in Figure 3, where three cloudlets are connected to a gateway that provides access to a cloud-based data center. For illustration purposes, let us assume that Cloudlet 2 is located in a city center and it supports a number of smart cameras that run the VSOT application. Let us also assume the other two cloudlets are along a path between the suburbs and the city center. During rush hours (for instance), users usually move towards the city center, and thus Cloudlet 2 will receive more load from incoming users. In our scenario, this load is characterized by EEGTBG players with their smart phones.

If many EEGTBG players move to the region where Cloudlet 2 is located, it can become overloaded (processor, storage, and/or networking). Ideally, Cloudlet 2 should have enough capacity to handle the load from all its users, including the additional EEGTBG users that move towards its region. However, during many hours of the day this cloudlet would be underutilized by the VSOT application alone, which would mean low resource utilization and unnecessary energy consumption, in addition to potential operational and maintenance cost. To avoid such waste in resource use, Cloudlet 2 can be planned to support demand of applications in the lower-delay class, while also running other applications whenever feasible. Therefore, resource management policies to allocate resources between cloudlets and cloud should be able to handle variable demand whilst taking application characteristics and users mobility into account.

Allocation Policies for Fog Computing

Different application requirements along with the mobile nature of fog users calls for enhanced policies that can optimize computing resource utilization and offer quality of service accordingly. We illustrate how three different scheduling strategies can impact applications quality of service in a fog – namely the *Concurrent*, the *First Come-First Served* (FCFS), and the *Delay-priority* strategies. The comparison is intended to show how the prioritization of low delay applications (delay-priority) would improve application execution when compared to standard resource sharing techniques (concurrent and FCFS). These algorithms should run in a cloudlet when a new application request arrives, deciding where this request should run: in the cloudlet or in the cloud. This decision can be based on both the current cloudlet load (e.g. CPU) and the request requirements (e.g. delay), as detailed below

- *Concurrent strategy*: application requests that arrive at a cloudlet are simply allocated to such cloudlet, regardless of capacity or current usage.
- *First Come-First Served (FCFS) strategy*: requests are served in the order of their arrival, until there are no more computing resources available. For the sake of simplicity, we only consider CPU capacity, but multi-criteria decision-making is often considered in the scheduling literature. When the cloudlet becomes full (i.e., remaining CPU capacity is smaller than application requirements), applications start to be scheduled for execution at the cloud data center.
- *Delay-priority*: applications requiring lower-delay are scheduled first; the next class of application requests is scheduled in the cloudlet until there is no CPU capacity available, and the remaining applications are scheduled in the cloud.

The strategies implement a module merging mechanism as described in the Edge-ward placement algorithm,[11] where modules of the same application are grouped to be placed at the same device. In this merging mechanism, when a module is moved to another device, all modules of the same kind are also moved to the same device.

Table 1 Maximum CPU requirements (in millions of instructions per second [MIPS]) estimated for each applications model

VSOT				EEGTBG		
Object detector	Motion detector	Object tracker	User interface	Client	Concentration calculator	Coordinator
550	300	300	200	200	350	100

Evaluation Setup

To evaluate the different strategies in a fog computing scenario, we carried out simulations using iFogSim.[11] The iFogSim was chosen because it runs on top of the well-established CloudSim simulator, which has been widely utilized and tested, and it is a simulator that allows the hierarchical composition of edge devices, cloudlets, and clouds, also supporting the measurement of application delays. The evaluation aims at analyzing the performance of the applications in terms of delay as well as assessing how allocation policies influence network traffic. We have set up the scenario depicted in Figure 3 with four VSOT application instances running in Cloudlet 2 and twelve EEGTBG application users in cloudlets 1 and 3. Initially, six EEGTBG users are playing the game in locations close to Cloudlet 1 and other six players are closer to Cloudlet 3. We move the EEGTBG players one by one to Cloudlet 2, emulating a mobility behavior, in order to assess any quality of service degradation resulting from poor resource allocation. Since EEGTBG has low-latency requirements, we consider that a player in a cloudlet only plays against players in the same cloudlet. Results shown focus on the analysis of Cloudlet 2, which receives applications from moving users.

Each cloudlet had a processing capacity of 4,000 MIPS (millions of instructions per second) and was connected to the gateway through a link with 10,000 Kbps bandwidth and four milliseconds latency. The link between the gateway and the cloud had 10,000 Kbps bandwidth and 100 milliseconds latency. Mobile and camera devices were connected to the cloudlets through a link with 10,000 Kbps bandwidth and two milliseconds latency.

The scheduling decision-making takes place before the application executes, thus the actual CPU capacity used by each application module is not precisely known at scheduling time. At scheduling time, strategies must check if a cloudlet has enough free CPU capacity to handle each application module. In the scenario we considered, each module needed, during execution, at most the CPU capacity shown in Table 1.

During execution, each application uses at a given time CPU capacity that depends on the interaction between its modules (i.e., how much data it receives from/sends to other modules, which triggers CPU-intensive actions). These estimations come from the application description, which models the application as a directed graph, or workflow, with its attributes, as commonly found in the scheduling literature.[11,12] Precise estimation, however, is a challenging issue and it is a focus of research per se. For some applications, input characteristics (e.g., video or image quality) and historical data are good indicators to estimate future performance, while other applications exhibit unpredictable behavior and estimating their future performance does not provide very accurate results.

Results

Each application has a processing loop among its modules that must be accomplished to display the results. The *loop delay* is the time taken for an application loop to execute. In the VSOT application this loop starts with the camera sensors producing the video stream, goes through the motion detector, object detector, object tracking, and finally PTZ control, measuring how long it takes for a object to be detected, tracked, and the camera adjusted to have better images of this object. In the EEGTBG application, the loop comprises the EEG sensor transmission to the mobile phone, the client selection of consistent readings, the concentration calculator, the client again, and the display. This measures the time taken between the sensed concentration level and the display of the current game status to the user.

Figure 4 shows the delay of the VSOT and EEGBTG application loops for the three different scheduling strategies according to the number of users that have moved from cloudlets 1 and 3 to Cloudlet 2. When only one EEGBTG player has moved to Cloudlet 2, all three strategies have the same results for both applications. When the second player moved to Cloudlet 2, different scheduling strategies start to impact the applications differently. VSOT loop is delayed with the Concurrent and Delay-priority strategies, while maintaining a low delay with the FCFS strategy. This means that scheduling using the Concurrent strategy is bringing resource

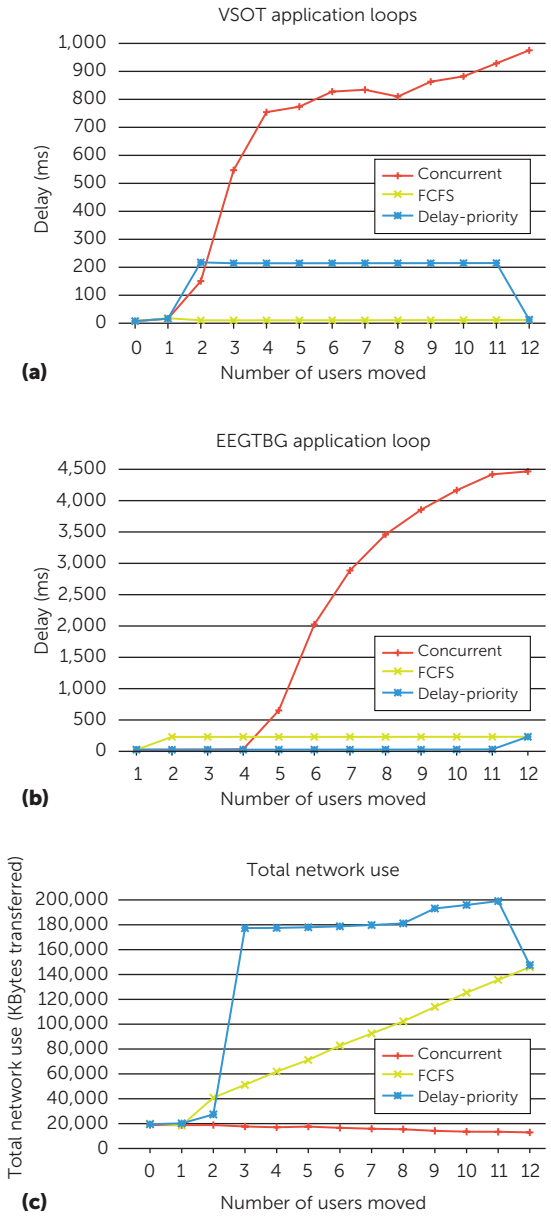

FIGURE 4. Application loop delays (*a* and *b*) and network usage (*c*) according to the scheduling strategy.

contention to Cloudlet 2, with the VSOT application experiencing a very high delay in its control loop, resulting in quality of service degradation and consequent application misbehavior, which is avoided by FCFS by moving EEGTBG modules to the cloud. On the other hand, the Delay-priority moves VSOT modules to the cloud as a result of the higher-priority EEGTBG application arrival, thus increasing the VSOT loop delay to an acceptable level of about 200 milliseconds while maintaining the EEGTBG delays as lower as possible.

The scheduling decision in fogs impacts in the total network use. Figure 4 also shows the total amount of data transmitted in the network for the different scheduling strategies. The Concurrent strategy results in lower network use, as it maintains all modules in the cloudlet and no communication between the cloudlet and the cloud is necessary. However, this comes at the expense of application delays, as discussed before. The FCFS strategy results in an increase of network use as EEGTBG players arrived at Cloudlet 2 and their modules are moved to the cloud. The Delay-priority strategy results in increased network use as it moves VSOT, a more network-intensive application, to the cloud. When 12 players arrive, the VSTO application is moved back to the cloudlet and the network use is reduced.

The Delay-priority strategy is effective in providing reduced delays for applications in the lower-delay class. However, when the 12*th* user moves to Cloudlet 2, the cloudlet does not have enough free CPU to handle all users. As a result, the Delay-priority strategy moves the EEGTBG modules to the cloud, increasing its delay. At this moment, the VSOT application is moved back to the cloudlet, thus presenting lower delays again. In this case, Cloudlet 2 has not enough capacity to handle the low-delay demand it is subject to, and therefore a cloudlet with more resources would be needed to avoid quality of service degradation.

Figure 5 shows the number of application modules scheduled on each device with the three different strategies. The Concurrent strategy simply increases the number of EEGTBG modules in the Cloudlet 2 as players arrive, maintaining all VSOT modules in the cloudlet as well, and this is reflected in the increasing in application loop delays previously presented in Figure 4. The FCFS strategy also maintains all VSOT modules in the cloudlet, but EEGTBG modules are moved to the cloud after the second player arrival to Cloudlet 2. The Delay-priority strategy has a more complex behavior. It starts moving VSOT modules from the cloudlet to the cloud when the second EEGTBG player arrives, and after that all VSTO modules stay in the cloud until the 12*th* EEGTBG player arrives. At this time, EEGTBG modules are moved to the cloud because Cloudlet 2 does not have enough CPU capacity to handle all users, and then VSOT is moved back to Cloudlet 2. Meanwhile, the number of EEGTBG modules in the cloudlet increases with the players arrival up to the 8*th* user arrival. When the 9*th* player arrives at Cloudlet 2, the Delay-priority strategy detects there is no room for all 18 application modules (i.e., nine concentration calculators plus nine

coordinators), so it groups concentration calculators in the cloudlet and coordinators in the cloud. When the 12*th* player arrives, the cloudlet is not able to handle all them at the same time: all EEGTBG modules are then sent to the cloud, and the VSOT application is moved back to the cloudlet.

Challenges and Future Directions

Fog computing brings challenges at many levels, starting from cloudlet placement, ownership, and business model. As soon as cloudlets are deployed, they bring many new interesting challenges to resource allocation and scheduling. Among those, we consider *application classification* and *user mobility* as two key aspects to be associated with scheduling in providing efficient resource management.

Application classification must provide the scheduler with information about application requirements, which will allow the scheduler to prioritize the cloudlet use and optimize other objectives (e.g., reduce network use, reduce cloud costs). With that information, a fog scheduler can decide which application should run in the cloudlet and which should run in the cloud. Moreover, application classes could also allow a system-level scheduler to prioritize applications within a cloudlet, allowing smaller granularity control over the delays observed by applications at each class.

Understanding users' behavior and mobility patterns can improve resource management by better planning the applications scheduling beforehand. This planning is crucial to avoid application delays during user movement. For example, in the scenario discussed in this paper the VSOT application must be moved from Cloudlet 2 to the cloud when EEGTBG users arrive. If a predictive mechanism is able to accurately determine when this migration should start, the VSOT application can experience lower delays for as long as possible, and the EEGBTG players would not experience larger delays if VSOT is moved only after their arrival at Cloudlet 2. Note that this planning can also involve data movement, depending on the application being migrated. In this case, planning should also consider the time taken to move data between cloudlets or from cloudlets to the cloud (and vice-versa).

Although mobility can be reasonably predicted in general,[13] prediction misses will eventually occur from lack of information or user unpredictable behavior. Scheduling strategies to deal with mobility prediction failure are also an interesting problem to be studied in the fog computing context. Different application classes may benefit from different strategies to work around mobility prediction failures. Also, users can be classified at different predictability levels to ensure the right strategies are applied to each one of them. Moreover, uncertainty in bandwidth availability and application modules processing times can also be expected to occur, which is also a challenging scheduling issue to be taken into account.

With application classes and mobility patterns, scheduling models that capture such characteristics can be developed and more efficient resource management algorithms can be designed. However, even with a theoretically efficient scheduling algorithm for fog computing in hand, resource management

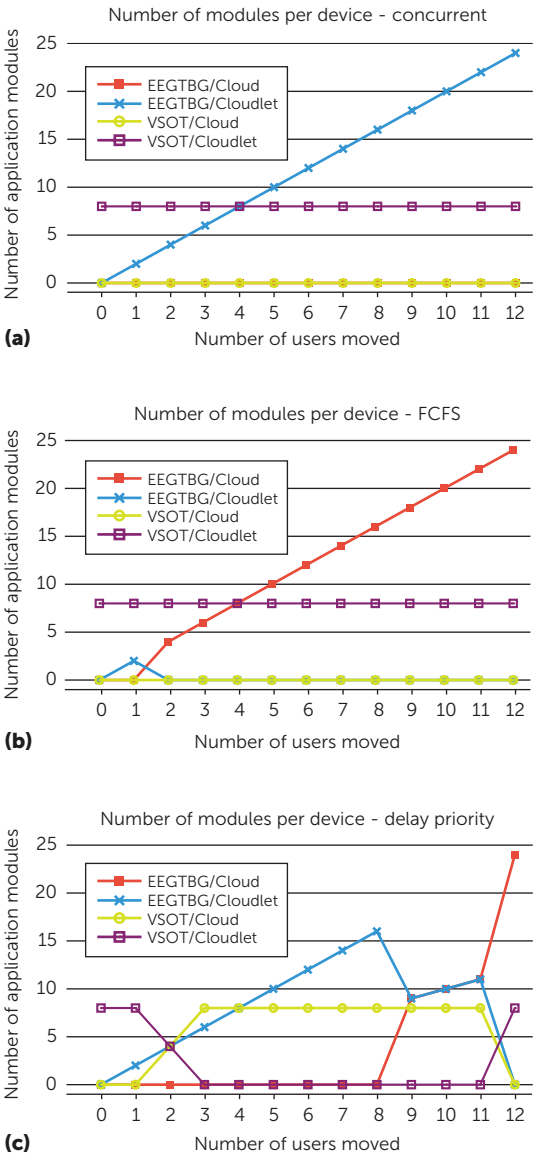

FIGURE 5. Number of instances scheduled at each device according to the scheduling strategy.

deployment still faces challenges due to uncertainties generated by the dynamicity and heterogeneity of the resources composing the infrastructure. Given a fog infrastructure with its cloudlets, a set of users with their predicted paths, and the resulting scheduling that optimizes a pre-defined objective function, mechanisms that can handle application deployment, movement, and resource reservation must be implemented. Resource virtualization is one promising way of dealing with application/data movement for each user in an isolated way. A possible implementation of such mechanisms for resource configuration, allocation, and reservation could involve computing virtualization tools such as virtual machines (VMs) or containers, as well as networking tools such as networking virtualization and software defined networks (SDN). Moreover, maintaining connectivity without service disruption while migration occurs is another interesting and challenging aspect, as proposed by the FollowMe Cloud.[14]

Application execution costs in a fog utility model are also interesting areas to explore. Given a business model for the cloudlets (how service levels agreements are offered – how cloudlets are commercialized and charged), schedulers should take into account a trade-off between costs and application quality of service. Scheduling algorithms for hybrid clouds such as the HCOC[12] could be extended to consider the fog computing hierarchy. Moreover, costs and delays of both storage data transfers from/to cloud providers can also take part in the trade-off.

The combination of advanced scheduling techniques, supported by applications classification and mobility prediction, with virtualization tools within an autonomic computing framework, such as CometCloud,[15] would be able to handle the dynamic mobile environment of a fog infrastructure and its clients.

Summary and Conclusions

Fog computing provides lower communication latencies and computing capacity closer to the final user. For this infrastructure to become efficient and offer actual differentiated service from the cloud computing paradigm, proper resource management mechanisms must be deployed.

In this paper we introduced the scheduling problem in the hierarchical composition of fog and cloud computing. The dynamic scenario resulted from users mobility brings a dynamic computing demand at edge devices, herein the cloudlets, from a variety of applications classes with particular requirements. We show that scheduling strategies can be designed to cope with different application classes according to the demand coming from mobile users, taking advantage of both the fog proximity to the end user and the cloud computing elastic characteristic. We also discussed challenges that arise from the mobile, dynamic fog users behavior, raising central research points that can be addressed in fog computing resource management.

Acknowledgments

We would like to thank Harshit Gupta for his support with iFogSim. LFB acknowledges the support from grant #2015/16332-8, São Paulo Research Foundation (FAPESP), European Commission H2020 programme under grant agreement no. 688941 (FUTEBOL), as well from the Brazilian Ministry of Science, Technology and Innovation (MCTI) through RNP and CTIC, CNPq and CAPES. This research is supported in part by NSF via grants ACI 1339036, ACI 1441376. The research at Rutgers was conducted as part of the RDI.[2]

References

1. F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*. plus 0.5em minus 0.4em Springer International Publishing, 2014, pp. 169–186.
2. L. F. Bittencourt, O. Rana, and I. Petri, "Cloud computing at the edges," in *Cloud Computing and Services Science*, M. Helfert, V. Méndez Muñoz, and D. Ferguson, Eds. plus 0.5em minus 0.4emSpringer International Publishing, 2016, pp. 3–12.
3. B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, 2015.
4. S. S. Iyengar and R. R. Brooks, *Distributed sensor networks: sensor networking and applications*. plus 0.5em minus 0.4emCRC press, 2016.
5. H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.
6. K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, ser. MCC '13. plus 0.5em minus 0.4emNew York, NY, USA: ACM, 2013, pp. 15–20.
7. M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the internet of things," *IEEE*

Pervasive Computing, vol. 14, no. 2, pp. 24–31, Apr. 2015.
8. A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments: Challenges, taxonomy, and survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 7:1–7:47, May 2014.
9. M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-edge Computing (MEC) industry initiative*, 2014.
10. 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 *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 945–953.
11. H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments," *CoRR*, vol. abs/1606.02007, 2016.
12. L. F. Bittencourt, E. R. M. Madeira, and N. L. S. Da Fonseca, "Scheduling in hybrid clouds," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 42–47, 2012.
13. C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.
14. T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, Sept. 2013.
15. J. Diaz-Montes, M. AbdelBaky, M. Zou, and M. Parashar, "Cometcloud: Enabling software-defined federations for end-to-end application workflows," *IEEE Internet Computing*, vol. 19, no. 1, pp. 69–73, Jan. 2015.

LUIZ F. BITTENCOURT *is assistant professor at the University of Campinas (UNICAMP), Brazil. His research interests include resource management and scheduling in cloud and fog computing. Bittencourt holds a PhD in computer science from UNICAMP. He is IEEE, ACM, and SBC member.*

JAVIER DIAZ-MONTES *is Assistant Research Professor at Rutgers University and a member of the Rutgers Discovery Informatics Institute (RDI2). He received his PhD degree in Computer Science from the Universidad de Castilla-La Mancha, Spain (``Doctor Europeus'', 2010). Before joining Rutgers, he was Postdoctoral Fellow at Indiana University. His research interests are in the area of parallel and distributed computing and include autonomic computing, cloud computing, virtualization, and scheduling.*

RAJKUMAR BUYYA *is a Professor of computer science and software engineering, a Future Fellow of the Australian Research Council, and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He also serves as the founding CEO of Manjrasoft. His research interests include resource management and scheduling systems for utility-oriented computing systems. Buyya has a PhD in computer science from Monash University. He's a Fellow of IEEE.*

OMER F. RANA *is Professor of Performance Engineering at Cardiff University, UK. He leads the Complex Systems research group in the School of Computer Science & Informatics and is director of the "Internet of Things" Lab. He holds a PhD in Computer Science from Imperial College (London University), UK. He is a member of IEEE.*

MANISH PARASHAR *is Distinguished Professor of Computer Science at Rutgers University. He is also the founding Director of the Rutgers Discovery Informatics Institute (RDI2). His research interests are in the broad areas of Parallel and Distributed Computing and Computational and Data-Enabled Science and Engineering. Manish is founding chair of the IEEE Technical Consortium on High Performance Computing (TCHPC) serves on the editorial boards and organizing committees of a large number of journals and international conferences and workshops, and has deployed several software systems that are widely used. He has received a number of awards for his research and leadership. Manish is Fellow of AAAS, Fellow of IEEE/IEEE Computer Society and ACM Distinguished Scientist. For more information please visit http://parashar.rutgers.edu.*