Social Virtual Objects in the Edge Cloud

Ivan Farris, University Mediterranea of Reggio Calabria, Italy
Roberto Girau, University of Cagliari, Italy
Leonardo Militano, University Mediterranea of Reggio Calabria, Italy
Michele Nitti and Luigi Atzori, University of Cagliari, Italy
Antonio Iera, University Mediterranea of Reggio Calabria, Italy
Giacomo Morabito, University of Catania, Italy

In the social Internet of Things, cyber counterparts of physical objects—social virtual objects—located at the edge of the network can help reduce communication delays and inefficiency.

In the natural world, animals have developed sociality to efficiently and effectively face the threats of a complex environment. A network of relationships link individuals, who form (often overlapping) communities. They play specific roles in their communities and rely on other members to get what they need but can't obtain by themselves.

This is the underlying rationale that has led to the social Internet of Things (SIoT).[1] In the SIoT, objects establish social-like relationships with each other. The resulting network of relationships is structurally navigable, enabling searches for specific services in effective and efficient ways. Furthermore, by leveraging social relationships,

 2325-6095/15/$31.00

objects can rely on differentiated levels of trustworthiness, which could support security.[2] Finally, social relationships can be established between objects that use different technologies, which is important to support interactions between elements across different IoT platforms.

However, few IoT devices have the processing and communication capabilities to create and manage social relationships. As a means to address these issues, IoT cloud platforms represent an essential component for the continuous deployment, provisioning, and execution of IoT applications.[3] Accordingly, implementations of the SIoT model envision cyber counterparts of physical objects, which we call *social virtual objects* (SVOs), running on servers in the cloud. Merging social network features and cloud platforms in the context of the IoT is promising in the view of deploying a plethora of novel IoT services, such as in the intelligent transportation systems (ITS) domain.

A cloud-based platform for the SIoT paradigm must provide the following features:

- low-latency interaction between physical objects and their digital counterparts by reducing the round-trip time in the relevant connections with regard to remote cloud hosting;
- scalability to meet the tremendous increase of traffic generated by the expected huge number of IoT objects;
- autonomous social agents that help resource-constrained IoT objects maintain, update, and browse their social relationships in a self-consistent way;
- flexibility to enable dynamic service provisioning and composition by combining the capabilities of SVOs, also deployed in different cloud servers; and
- mobility management to provide proximity services based on a physical device's position by supporting seamless migration of SVOs across geographically distributed cloud servers.

To meet these challenges, we propose adopting edge cloud technologies in the implementation of the SIoT platform. Specifically, we present the changes introduced in a traditional cloud-platform-based implementation of the SIoT solution, consequent to the use of edge cloud technologies, which must be able to detect the need for a change in the geographical location of the virtual object and to handle the intercloud mobility of related processes and data. Analysis of a use case demonstrates the need for our proposed solution and demonstrates its advantages.

The Move toward Edge Cloud in the SIoT

In this section we first provide some background information about the integration of IoT, cloud and edge computing, then we present the basic concepts related to the Social Internet of Things.

Cloud and Edge Computing for the IoT

The recent integration of cloud computing features in the IoT landscape represents a major breakthrough in the deployment of sophisticated applications, requiring heavy processing not natively supported by resource-constrained IoT devices. In this context, data generated by IoT devices can be sent and processed in the cloud,[4] supported by the considerable computational and storage capabilities available in datacenters.

A further leap forward in this domain is represented by the introduction of solutions that envision the instantiation of digital representatives (virtual objects) of the physical objects in the cloud,[5] the cloud of things. In this way, added-value applications can be implemented by exploiting services offered by virtual objects. These applications can be developed independently of the specific hardware features of the IoT devices.

However, for applications with strict requirements in terms of delay, the latency caused by interactions with or between virtual objects in the cloud is unacceptable. An emerging trend leverages the edge cloud, which introduces intelligence and flexibility into network edge nodes able to host cloud applications, moving cloud services much closer to users' devices.[6] These applications can cooperate with and offload corresponding applications residing in users' smart objects and conventional cloud centers to offer innovative services. This will allow user requests to be manipulated before crossing the network toward datacenters in ways that enhance performance. Manipulations include preprocessing, decomposition, and proxying. This solution aligns

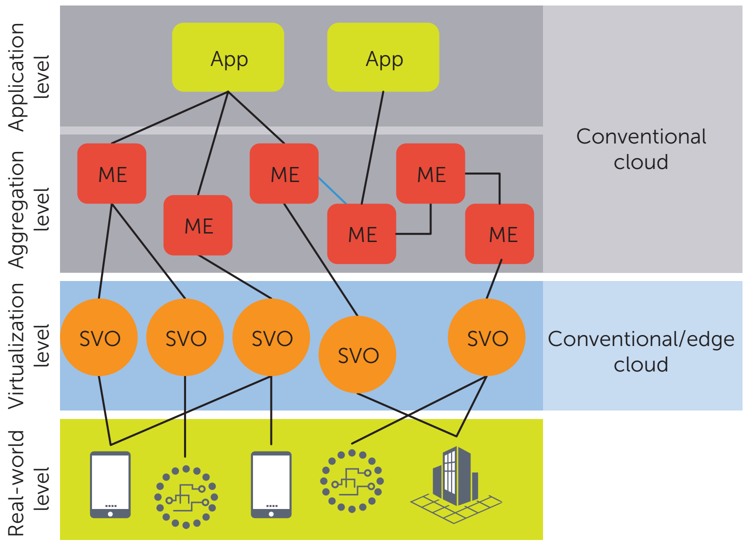

FIGURE 1. Cloud-based social Internet of Things (SIoT) architectural solution. The virtualization layer is the only one that can be implemented in both the conventional and the edge cloud. (ME: micro engine, SVO: social virtual object)

with the cloudlet and fog computing paradigms,[7] and is supported by European projects such as Tropic (www.ict-tropic.eu) and Input (www.input-project.eu).

According to the cloudlet and fog computing paradigms, network edge devices are evolving into microcloud servers, able to host not only advanced network functions but also application modules. This allows for instantiating distributed functionalities, accommodating the desired level of quality of experience (QoE) and workload/traffic volumes, closer to users. Furthermore, the distributed architecture composed of large-scale geographically deployed nodes is inherently scalable.

Social IoT Model

In the last few years, the academic community has focused on adding sociality at different levels of the IoT.[8,9] In this article, without losing generality, we refer to the SIoT model proposed in earlier work, in which different forms of socialization among objects might exist.[1] The *parental object relationship* (POR) is defined among similar objects built in the same period by the same manufacturer, where the production batch is considered a family. The objects can establish a *colocation object relationship* (C-LOR) or *cowork object relationship* (C-WOR), like humans do when they share personal (such as cohabitation) or public (such as work) experiences. We define a further type of relationship for objects owned by the same user: ownership object relationship (OOR). Finally, the social object relationship (SOR) is established when objects come into contact, sporadically or continuously, for reasons purely related to relations among their owners.

The SIoT architecture consists of four major components.[1] The *relationship management* functionality introduces into the SIoT the intelligence that allows objects to establish, update, and terminate relationships, according to rules set by their owners. The *service discovery* functionality determines which objects can provide the required service in the same way humans seek friendships and information. The *service composition* functionality enables interaction among objects, and *trustworthiness management*[2] aims to understand how the information provided by other members should be processed.

SIoT Cloud-Based Architecture

Researchers[3,4,5] and funded projects (for example, iCore and Compose projects) have investigated the use of virtual entities in the IoT to represent real-world objects. Lysis, our proposed cloud-based platform,[10] leverages results of these works and adds the social aspects defined by the SIoT paradigm. Its platform-as-a-service (PaaS)-oriented design offers a four-level architecture (see Figure 1):

- *Real-world level*. Implemented outside of the cloud, this level includes information and communications technology devices that can access the Internet. These real-world objects (RWOs) are directly connected to the physical environment where they sense and act.
- *Virtualization level*. Each object in the real-world level has a corresponding virtualization. Social virtual objects (SVOs) are representations of RWOs in terms of their semantic description and functionalities extended with social capabilities.
- *Aggregation level*. This level aggregates data from multiple SVOs on the basis of patterns to ensure high reusability. Micro engines (MEs) acquire and process data from SVOs into high-level services requested by applications.
- *Application level*. At this level, user applications perform final processing and presentation.

The SVO is the key part of the overall solution. An SVO is equipped with interfaces to establish a secure connection with the RWO and allow for a standardized communication procedure between the aggregation level and the variegate set of physical devices. It implements a variety of functions, such as service discovery,[1] which aims to find potential providers of information among SVOs in the

SIoT community. This distributed process is accomplished for each user by the user's SVO root (SVOR), which is selected from among all user SVOs. The SVOR accepts requests for SVO search from the application and aggregation levels and returns resource addresses by browsing objects' social graphs and the API keys to access them. We identify three permission levels: public, private, and friend. The last two levels require an owner or friend key, respectively.

Exploiting the Edge Cloud

To exploit the benefits of a distributed cloud service infrastructure and the fog computing paradigm, our platform leverages a distributed cloud solution and dynamically deploys SVOs closer to their physical counterparts.

The Lysis platform, implemented using the Google App Engine (GAE) PaaS, is the starting point of our research activity. GAE APIs let us implement a template repository of friends on each SVO using a document representation, enabling full-text search through the social graph. We also created a uniform repository of locations, which are needed to establish C-WOR and C-LOR social relations based on object positions.

The highlighted advantages of a distributed cloud paradigm imply a profound transformation of the Lysis platform. In fact, Lysis relies on a conventional cloud system, which assumes a permanent execution location of Web services localized in classic partitioning of availability zones and regions; thus, it doesn't acount for geographical distance between real objects and their virtual counterparts. To fully exploit the short distance between edge cloud and devices, supplementary architectural elements are necessary.

The *deployer* (see Figure 2) creates an SVO whenever the user registers a new device. The device owner triggers the creation of a new SVO by selecting the right template from the repository, so a new SVO instance is created in the container located in the owner cloud space. During this phase, the new SVO inherits the semantic description of its template, which is recorded in its profile stored in the owner database space. As soon as the SVO is created, the owner links it to the physical device, which sends the basic information. The user can add further descriptions with contextual information. Communication between SVOs is based on RESTful Web services. However, to set up the communication, the SVOs need to be friends, so that they know each other's URI and API friend key.

Figure 2 highlights the main requirements for SVO deployment and migration. In our platform, the SVO is a Web service that requires two basic elements

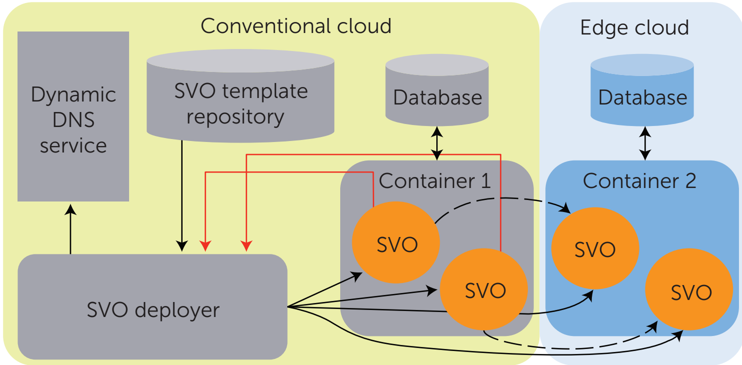

FIGURE 2. Elements at the virtualization level involved in the deployment and migration of social virtual objects (SVOs). The deployer manages the process of moving the source code and database information of the migrating SVOs to the edge cloud.

for its implementation: a container and a database. The container is an execution environment for Web services and has a role in both deployment (when users register new objects) and migration (when the SVO asks the deployer module for a migration).

When operating in the edge cloud, the first key component is a module able to monitor all parameters, such as quality of service (QoS), latency, and power consumption. These parameters represent the basis for deciding whether to migrate an SVO from the conventional (remote) cloud to the edge cloud. Whenever a migration is performed, the relevant DNS record needs to be updated. When an SVO detects that a migration must be performed, it sends a request to the deployer together with the history of sensor data and relationships that are stored in the database in the conventional cloud. Migrations are always performed after completion of the running process. Since the SVO is an event-driven system, its state with all the relevant information can be saved into the database after every event-triggered process. Thus the SVO migration process requires the deployer to put the SVO template into a container in the edge cloud and to make a database dump of all data associated with that SVO. The SVO data is copied into the container's database in the edge cloud. Finally, the new SVO IP address is registered in the DNS server and the old instance is disabled.

Communications between the physical object and its SVO are implemented through interdomain transmissions. Communication frequency depends on the applications that are running in the physical device and can be predicted based on installed applications and the user profile. SVO search process instances trigger interactions among SVOs until the potential service provider is found. If all the SVOs are in the conventional cloud, the transmission latency is limited. Otherwise, if some SVOs are already migrated to the edge, edge-to-cloud and edge-to-edge communications might be needed.

Exemplary Workflow for an ITS Use Case

To better describe our proposed platform's workflow, we focus on ITS scenarios, which have attracted the attention of both cloud and social computing research communities.[11,12]

In our ITS use case, Angela has bought a new smart car and the corresponding SVO has been added to her cloud of things in the platform. The registration phase includes the creation of the SVO using the suitable template, and installation of the appropriate hardware abstraction layer module on the physical device to allow it to interact with its digital counterpart. In this way, the SVO can provide access to physical devices' resources and begin to create social relationships with other SVOs. If additional resource-constrained devices are embedded in the vehicle, such as driver monitoring or environmental sensors, the car can act as a gateway for their SVOs in the SIoT framework, implementing short-range communications with the physical devices.

Angela browses the platform's online application store and selects the social-IoT-based navigation (SIT-NAV) system, which exploits object cooperation and resource sharing to help users select the optimal route to their destination. To plan her route, Angela can define her preferred rules or conditions for the trip. Accordingly, the platform deploys the SIT-NAV service with the relevant modules (MEs). Each ME requires input data, which can be provided by the available SVOs through service discovery.

Figure 3 depicts a sample scenario where at time $t = t_0$ Angela wants to find the best route to the airport and an estimated arrival time to be sure she can reach her flight on time. Cars, traffic lights, and other objects in the environment exchange relevant information through their friend SVOs at the edge node.

The SIoT is supported by the remote and edge clouds. The former represents the conventional cloud servers with significant resources, whereas the latter refers to micro datacenters deployed at the network's edge. Some SVOs might be located in the remote cloud, but most are located in the edge clouds for more efficient SVO updating and service discovery. Each ME in the SIT-NAV requires appropriate information to perform the relevant services and forwards the request to the user's SVOR (in our scenario, Angela's vehicle). By exploiting the social relationships, the SVOR obtains the addresses of the other SVOs that can provide the desired resources, and communicates them to the ME. The ME can then request or subscribe to the desired SVOs' resources. When the user and its associated devices move along the path, the serving edge node can change and an SVO migration can be triggered according to the SVO migration policies. As Figure 3 shows, at time instant $t = t_1$, the SVO for Angela's vehicle is moved to the second edge node, where having virtual images of physically nearby devices at the same edge node in the network increases the probability of obtaining the required service or data from colocated SVOs.

Performance Analysis

We conducted a numerical evaluation in Matlab for a wide set of scenarios to observe the achievable performance in terms of average number of messages (and relevant cost in terms of latency) exchanged by an object to communicate with its virtual counterpart when moving the SVOs from the cloud to the edge node.

The cited performance analysis would need information about the position and mutual relationships of a large number of objects. This data isn't available to date, as real applications have not been deployed yet. We therefore used the real dataset of the location-based online social network Brightkite obtained from the Stanford Large Network Dataset Collection[13] and extended it to account for social relationships. We assume that every person carries one smart object, such as a smartphone, so when they get in touch with their friends, their objects also come into contact and can create an SOR.

In addition, C-WOR, C-LOR, and POR relationships are created based on the mobility of the humans carrying their objects. The resulting SIoT network has around 5k nodes and 36k edges.

As described previously, SVOs mainly exchange two types of messages:

- *status update*, where physical objects send the monitored data of their surroundings and commands to their SVO, or vice versa; and
- *service discovery*, where SVOs crawl their social network of friends to find an SVO that can provide the requested service.

In our setting, we consider that every node updates its status with a frequency randomly chosen from the interval [6, 60] updates/hour. Each interaction among pairs of objects begins by randomly choosing an SVO client and, subsequently, selecting the SVO that can provide the requested service according to the geographic distance among them by

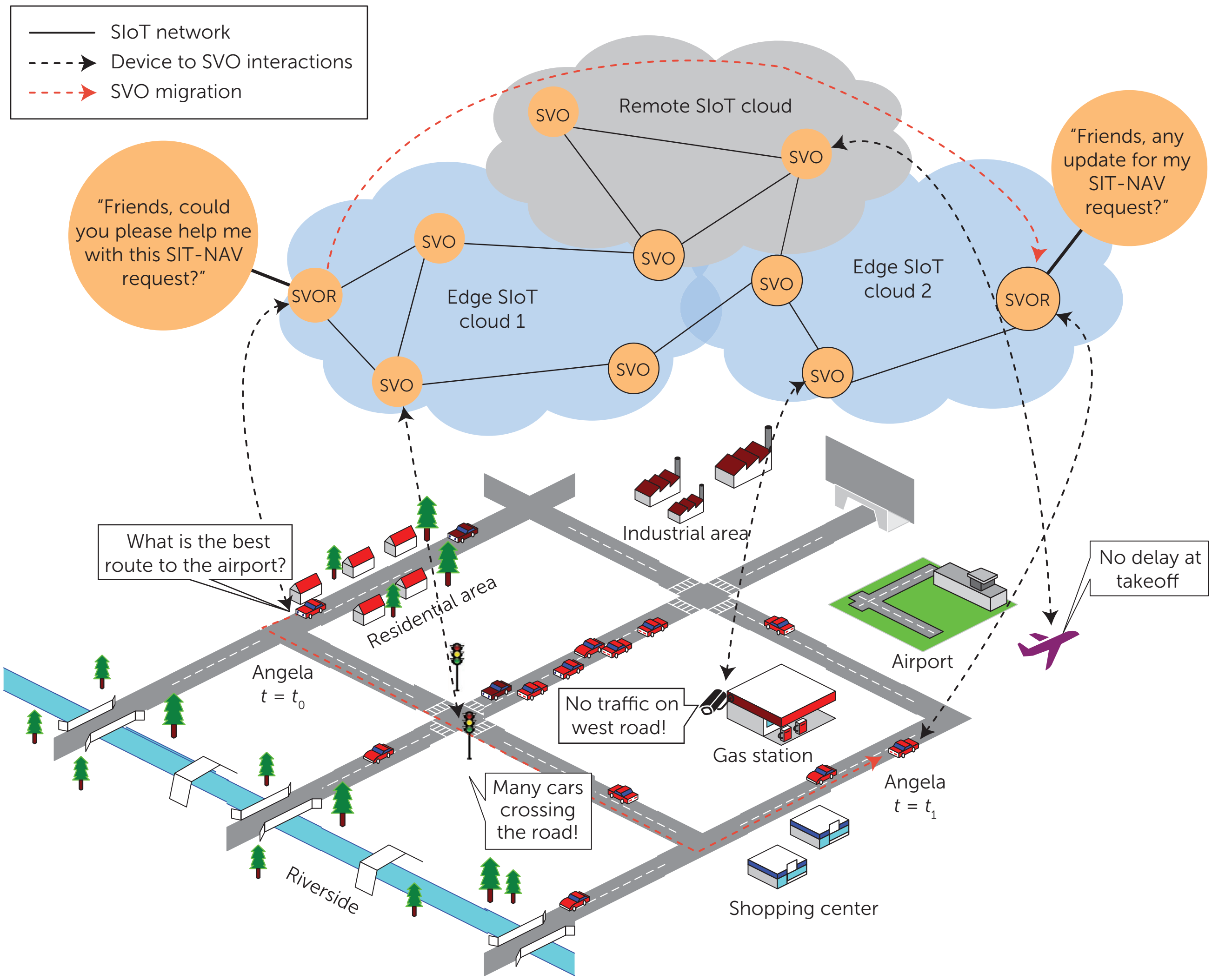

FIGURE 3. Example scenario for the social-IoT-based navigation (SIT-NAV) application. Cars, traffic lights, and other objects in the environment exchange relevant information through their friend social virtual objects (SVOs) deployed at the edge node.

following an exponential distribution. This models the location-based interest of the information, which is especially true in the ITS use case we're considering.[14] We adopt a migration strategy based on how often physical objects update their status; in particular, we migrate the SVOs in descending order of their status update frequency.

On the basis of these preliminaries, we evaluated the effectiveness of the SVO migration in terms of number of exchanged messages. Regarding the status update, communications take place between physical and virtual objects, and then, depending on the SVO's location, we can have either physical-cloud (labeled P-C in the graph) or physical-edge messages (P-E). During the SVO search process, we also identified four message types:

- intracloud messages (intra-C), when all the SVOs involved in the SVO search process are in the cloud;
- edge-cloud messages (E-C), when an SVO in the edge node communicates with an SVO in the cloud, or vice versa;
- intraedge messages (intra-E), when communications between SVOs occur in the same edge node; and
- interedge messages (inter-E), when more than one edge node is involved.

Figure 4a shows the ratio of each type of message in several scenarios, and provides results for different values of the percentage of migrating nodes. For each percentage value there are two bars: the one

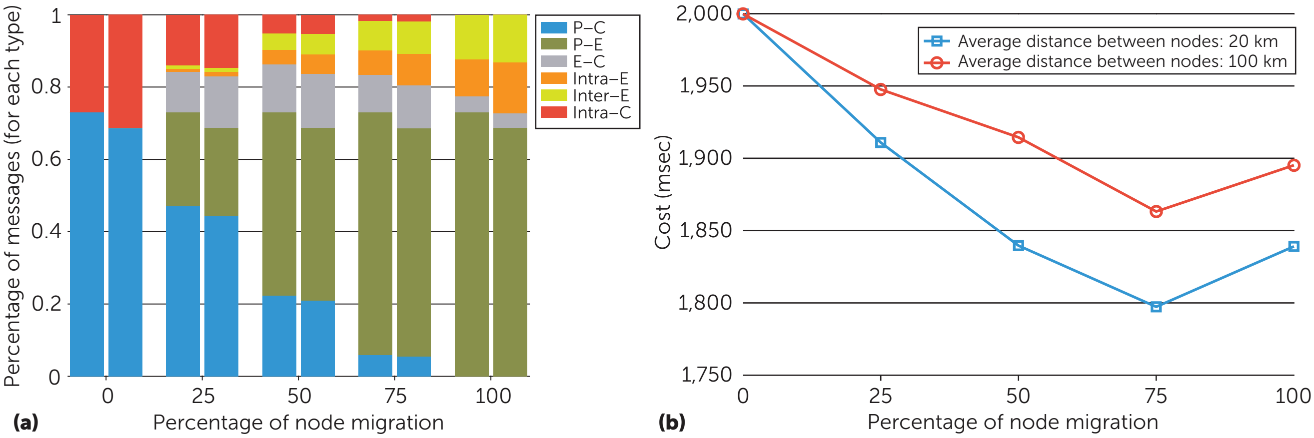

FIGURE 4. Results from analysis: (a) breakdown of the exchanged messages and (b) cost of the message exchange versus the percentage of migrating nodes for different distances between interacting objects.

on the left is obtained when the interacting objects are 20 km distant from each other, and the one on the right is obtained when the distance is 100 km. A larger distance results in a slightly higher number of hops needed to navigate the network and, therefore, a higher percentage of messages relevant to service discovery with regard to status update. This effect can be observed when all the SVOs are in the cloud and there are only P-C and intra-C messages. In fact, the frequency of intra-C messages increases, thus lowering the P-C messages' frequency. Moving the SVOs from the cloud to the edge leads to a reduction of P-C and intra-C messages, which eventually are no longer involved when all the SVOs are migrated, and to a growth in the number of messages involving the edge. In particular, physical objects will update their status on the SVO in the edge node and the SVO search process will only involve SVOs in the edge nodes. However, the cloud is still involved in some communications, such as E-C messages, when the application located in the cloud sends a request for an SVO search to the SVOR. It's worth noting that, even with a complete migration, the difference in the number of inter-E messages between the 20- and 100-km scenarios is limited. This limited difference is achieved following the navigation mechanism of the social networks. In fact, long distances are usually travelled in no more than one or two hops, and a fine-grained search is then performed locally.[15]

However, messages don't have the same cost in terms of latency. To account for their relative importance, we focused on a generic scenario and calculated the latency for each message type over several runs, considering different cloud providers and edge nodes. The obtained latency of P-C messages is equal to 60 msec, which corresponds to the sum of the costs of P-E and E-C messages, which are 42 and 18 msec, respectively. Messages between edge nodes (inter-E messages) are the most onerous if you consider the worst case—that is, when they have to traverse low-speed routers belonging to different service providers with regard to routers in the Internet core. These messages have a latency of 72 msec. Nevertheless, the latency of inter-E messages represents an upper bound since messages traveling from one edge node to another belonging to the same telco operator have a latency similar to the E-C messages. Finally, intra-E and intra-C messages have a negligible cost in terms of latency. Figure 4b shows the resulting graph, which compares the use of the conventional cloud and that of the proposed framework for different percentages of node migration. In particular, when the percentage is 0 (such as in the conventional cloud scenario), we obtain the highest value for the latency. The latency decreases when we migrate the SVOs to the edge nodes. However, a complete migration (100 percent) isn't desirable because it would lead to higher latency costs due to the high number of inter-E messages. The design of the migration algorithm is important to understanding which SVOs must be instantiated in the edge node; thus, this proof-of-concept analysis requires further investigation.

Finally, note that we didn't take into account the *una tantum* cost for migrating from the cloud to the edge node. However, because of the mobility of users and their associated physical objects, we might need to migrate an SVO from one edge node to another. In this case, additional design choices must be considered.

Our preliminary analysis emphasized the need for strategies to dynamically decide which SVOs to migrate from cloud to edge, and vice versa. Further research directions might focus on the efficient management and exploitation of social relationships for SVOs. The selection of a good set of social relationships will influence the resource searching process through the social graph and, consequently, affect our propsed solution's performance.

On the other hand, further architectural enhancements might be considered to enable the proposed approach. Specifically, microtasks could be directly injected into physical devices to better exploit their growing computational capabilities at the cost of introducing a proper PaaS functionality inside the device firmware.

Finally, an important issue to be addressed is the trust of the edge resources where SVOs are hosted. Indeed, trustworthiness of the cloud and of the network edge facilities is taken for granted in our scenario. The edge cloud should be managed by the network operator that the objects are using for data connection services. As we increasingly encounter encryption of content provided by over-the-top service providers (notwithstanding the increase in traffic overhead), we can imagine that the information associated with SVOs that moved from the conventional cloud to the edge cloud will be encrypted as well. In this way, this information (for example, collected sensed data, object profiles, or object usage statistics) will remain private. How the encryption will be performed by the physical devices with limited resources is, however, still an open issue. Additionally, trust in the reliability of the computing and storage services at the network edge in terms of availability and retainability will remain an issue. We expect that QoE and the relevant impact on customer churn will force network providers to consider reliability as critical in retaining market share.

Acknowledgments

This work was partially supported by the Input (In-Network Programmability for next-generation personal cloUd service supporT) project funded by the European Commission under the Horizon 2020 Programme (Call H2020-ICT-2014-1, Grant no. 644672) and by Telit Communications SpA in project "Platform for the deployment of distributed applications in Vanets and WSNs."

References

1. L. Atzori et al., "The Social Internet of Things (SIoT)—When Social Networks Meet the Internet of Things: Concept, Architecture and Network Characterization," *Computer Networks*, vol. 56, no. 16, 2012, pp. 3594–3608.
2. M. Nitti, R. Girau, and L. Atzori, "Trustworthiness Management in the Social Internet of Things," *IEEE Trans. Knowledge and Data Eng.*, vol. 26, May 2014, pp. 1253–1266.
3. H.-L. Truong and S. Dustdar, "Principles for Engineering IoT Cloud Systems," *IEEE Cloud Computing*, vol. 2, no. 2, 2015, pp. 68–76.
4. P. Parwekar, "From Internet of Things towards Cloud of Things," *Proc. Int'l Conf. Computer and Comm. Technology* (ICCCT), 2011, pp. 329–333.
5. S. Distefano, G. Merlino, and A. Puliafito, "Enabling the Cloud of Things," *Proc. Int'l Conf. Innovative Mobile and Internet Services in Ubiquitous Computing* (IMIS), 2012, pp. 858–863.
6. M. Ryden et al., "Nebula: Distributed Edge Cloud for Data Intensive Computing," *Proc. IEEE Int'l Conf. Cloud Eng.* (IC2E), 2014, pp. 57–66.
7. F. Bonomi et al., "Fog Computing and Its Role in the Internet of Things," *Proc. 1st Ed. MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16.
8. A. Ortiz et al., "The Cluster between Internet of Things and Social Networks: Review and Research Challenges," *IEEE Internet of Things J.*, vol. 1, June 2014, pp. 206–215.
9. P. Mendes, "Social-Driven Internet of Connected Objects," *Proc. Interconnecting Smart Objects with the Internet Workshop*, 2011.
10. R. Girau, S. Martis, and L. Atzori, "A Cloud-Based Platform of the Social Internet of Things," *Proc. EAI Int'l Conf. Cyber Physical Systems, IoT and Sensors Networks*, 2015.
11. A. Vegni and V. Loscri, "A Survey on Vehicular Social Networks," *IEEE Comm. Surveys Tutorials,* vol. 17, no. 4, 2015, pp. 2397–2419.
12. M. Whaiduzzaman et al., "A Survey on Vehicular Cloud Computing," *J. Network and Computer Applications*, vol. 40, Apr., 2014, pp. 325–344.
13. J. Leskovec, "Stanford Large Network Dataset Collection," 2015; http://snap.stanford.edu/data.
14. M. Gerla et al., "Internet of Vehicles: From Intelligent Grid to Autonomous Cars and Vehicular Clouds," *Proc. IEEE World Forum Internet of Things* (WF-IoT), 2014, pp. 241–246.
15. M. Boguna, D. Krioukov, and K.C. Clay, "Navigability of Complex Networks," *Nature Physics*, vol. 5, no. 1, 2009, pp. 74–80.

IVAN FARRIS *is a PhD student in information technology engineering at the University of Reggio Calabria, Italy. His research interests include wireless networks, Internet of Things, and mobile cloud computing. Farris has an MSc in computer and telecommunications systems engineering from the University of Reggio Calabria. Contact him at ivan.farris@unirc.it.*

ROBERTO GIRAU *is a researcher in the Department of Electrical and Electronic Engineering of the University of Cagliari. His research interests include the Internet of Things, with particular emphasis on its integration with social networks, and software engineering. Girau has an MSc in telecommunication engineering from the University of Cagliari, Italy. Contact him at roberto.girau@diee.unica.it.*

LEONARDO MILITANO *is an assistant professor in the Department of Information Engineering, Infrastructure and Sustainable Energy at the Mediterranea University of Reggio Calabria, Italy. His research interests include wireless network optimization, radio resource management, device-to-device communications, game theory applications to networking problems, and user cooperation. Militano has a PhD in telecommunications engineering from the University of Reggio Calabria. Contact him at leonardo.militano@unirc.it.*

MICHELE NITTI *is an assistant professor in the Department of Electrical and Electronic Engineering at the University of Cagliari, Italy. His research interests include the Internet of Things, particularly the creation of a network infrastructure to allow the objects to organize themselves according to a social structure. Nitti has a PhD in electronic and computer engineering from University of Cagliari, Italy. Contact him at michele.nitti@diee.unica.it.*

LUIGI ATZORI *is an associate professor in the Department of Electrical and Electronic Engineering at the University of Cagliari, Italy. His research interests include multimedia communications and computer networking (wireless and wireline), with emphasis on multimedia quality of experience (QoE), multimedia streaming, next-generation network service management, service management in wireless sensor networks, and architecture and services in the Internet of Things. Atzori has a PhD in electronic and computer engineering from University of Cagliari, Italy. He's a senior member of IEEE. Contact him at l.atzori@ieee.org.*

ANTONIO IERA *is a full professor of telecommunications and director of the Laboratory for Advanced Research into Telecommunication Systems at the University of Reggio Calabria, Italy. His research interests include next-generation mobile and wireless systems, RFID systems, and Internet of Things. Iera has a PhD in multimedia communication technology from the University of Calabria, Italy. He's a senior member of IEEE. Contact him at antonio.iera@unirc.it.*

GIACOMO MORABITO *is an associate professor in the Department of Electrical and Electronic Engineering and Informatics at the University of Catania, Italy. His research interests include analysis and solutions for wireless networks and Internet of Things. Morabito has a PhD in electrical, computer, and telecommunications engineering from the Istituto di Informatica e Telecomunicazioni, University of Catania, Italy. Contact him at giacomo.morabito@dieei.unict.it.*