The Fog Makes Sense: Enabling Social Sensing Services With Limited Internet Connectivity

Ruben Mayer
Institute for Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
ruben.mayer@ipvs.uni-stuttgart.de

Harshit Gupta, Enrique Saurez
Umakishore Ramachandran
Georgia Institute of Technology
Atlanta, Georgia, USA
harshitg,esaurez,rama@gatech.edu

ABSTRACT

Social sensing services use humans as sensor carriers, sensor operators and sensors themselves in order to provide situation-awareness to applications. This promises to provide a multitude of benefits to the users, for example in the management of natural disasters or in community empowerment. However, current social sensing services depend on Internet connectivity since the services are deployed on central Cloud platforms. In many circumstances, Internet connectivity is constrained, for instance when a natural disaster causes Internet outages or when people do not have Internet access due to economical reasons. In this paper, we propose the emerging Fog Computing infrastructure to become a key-enabler of social sensing services in situations of constrained Internet connectivity. To this end, we develop a generic architecture and API of Fog-enabled social sensing services. We exemplify the usage of the proposed social sensing architecture on a number of concrete use cases from two different scenarios.

CCS CONCEPTS

•**Computer systems organization** →**Distributed architectures;** *Sensor networks;*

KEYWORDS

Social Sensing, Fog Computing, Situation Awareness

1 SOCIAL SENSING

Situation-aware applications use data streams from sensors to provide useful services to users or other applications. With the proliferation of sensors deployed in the surrounding world, e.g., through the Internet of Things, the potential of such applications is reaching new dimensions. Recently, research focus has been expanded from traditional fixed sensor deployments toward *social sensing* [1]. This comprises passive sensors provided by human carriers in Smart Phones, active human sensor operators taking pictures or videos and even humans operating as sensors themselves, e.g., providing live information in tweets and postings. Recently, new applications have been proposed which use the social sensing infrastructure to infer situations that are not detectable from traditional sensors.

1.1 Application Fields

An important application field of social sensing is in helping people to deal with natural disasters. There are applications that help in finding friends and family in the aftermath of a natural disaster [19]. Furthermore, social media can provide access to relevant and timely information to individuals in affected regions [17]. Providing real-time information to disaster-affected people about the situation in the area can help them take mitigative actions, for instance moving contents located in a flood-prone ground floor to upper floor [2] to reduce the loss caused by the disaster. Social media has been an effective way of sharing this sort of crowd-sourced information and can be more accurate and meaningful than government predictions. Many proposals envision disaster-stricken people to perform social sensing tasks, like providing information about the level of inundation of roads in the event of a flood or tsunami. Such unstructured information would be mined by a social sensing application to extract relevant details and create a map of the affected area with important information. These maps can be used by government agencies to perform rescue operations [8]. Users can upload pictures of people with them, and social sensing applications apply face recognition algorithms on the pictures and let the friends and family of detected individuals know that they are safe.

In rural or economically under-served regions, social sensing helps in understanding socioeconomic processes [11] which can empower communities to better utilize their social capital[1] and enable self-organized governance. Public transportation in such regions leave much to be desired due to lack of consistency in schedules and infrastrual support, forcing passengers to wait for long periods of time. In well-served communities, infrastructual support (e.g., kiosks at bus stops operating on GPS data) provide timely information for the passengers. Social sensing services in such under-served regions could help gather information, e.g., when

---

[1]Social capital refers to the features of social organizations that facilitate coordination and cooperation for mutual benefit.

Figure 1: System Model

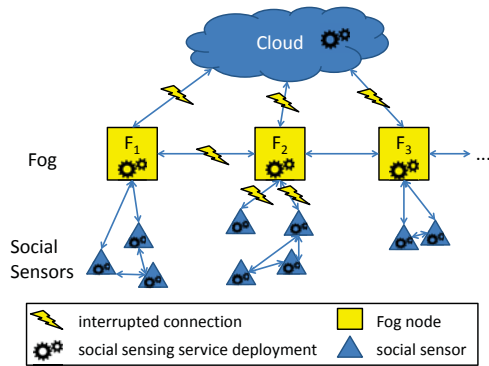| Device	Computational Capabilities	Containers Supported?	Connectivity
Routers	Low	Yes	WiFi, LAN
μComputers	Medium	Yes	WiFi, LAN
Drones	High	Yes	WiFi, 4G

Figure 2: Overview of Fog Devices.

the bus is going to arrive and share with others even in the absence of infrastructural support.

1.2 Challenges

While the discussed applications are very effective in utilizing social sensing information, they rely on Internet connectivity of the social sensors, the situation inference applications, and the users that are interested in the detected situations. This is mainly the case because the social sensing service is hosted in a central (cloud) data center.

However, Internet connectivity cannot be taken for granted on any of the layers of a social sensing application. Internet outages can affect large areas in case of emergencies, natural disasters, or hacker attacks on the Internet infrastructure [13]. Furthermore, rural regions might not be connected to the Internet at all, or the inhabitants of a rural or an under-served urban region cannot afford Internet connectivity for economical reasons. Social sensing applications can be of a huge benefit in exactly such situations and circumstances. All of those benefits are tightly coupled to the Internet connectivity; without the Internet, social sensing services are not available.

In recent years, a new trend has emerged in computing infrastructures that can help in overcoming the Internet dependency of social sensing services. *Fog Computing*, also known as Edge Computing, is the approach of adding computational resources toward the edge of the Internet [5]. While it was initially intended to improve network latency between sensors, applications, and users [10], we propose Fog Computing to become a central enabler of decentralized, local social sensing services that can also operate when Internet connectivity is constrained. This way, social sensing services can become more robust to Internet outages. Furthermore, communities that did not benefit from the first wave of cloud-based social sensing services can leapfrog those and directly use Fog-based services.

However, today's social sensing services are not capable of using the Fog infrastructure to provide local services when Internet connectivity is impaired. It is not enough to just run a centralized social sensing service on a number of Fog nodes in parallel. Instead, the social sensing service has to become a *distributed service* capable of discovering available Fog nodes and building a network that aggregates and shares information between social sensors that are connected to different Fog nodes. In this regard, it needs to be able to deal with the volatile nature of Fog and sensor connectivity.

To this end, the architecture of social sensing services needs to be adapted to fully utilize the opportunities of the Fog infrastructure.

1.3 Outline

In this paper, we give an overview of evolving Fog-based computing infrastructures. Based on that, we propose a generic architecture for Fog-based social sensing services. Using two concrete case studies, we demonstrate how existing cloud-based social sensing services can be adapted to use the Fog-based architecture. We conclude that utilizing Fog-based computing architectures is a promising path to more robustness and democratization of social sensing services.

2 FOG-BASED SOCIAL SENSING ARCHITECTURE

2.1 Overview

In the following, we give an overview of the emerging Fog Computing architecture. We point out that the Fog infrastructure can be completely heterogeneous. Social sensing on Fog has to be able to cope with the heterogeneity provided in the available resources.

Figure 1 shows a model of the Fog Computing architecture. On the top layer, the traditional Cloud data center is depicted, being deployed in the core of the network and only reachable via Internet connections. Such data centers are characterized by using standardized, off-the-shelf computing resources, and a virtualization layer that allows for an effective utilization of the resources and a pay-as-you-go business model. In the middle layer, a number of heterogeneous Fog nodes are geographically distributed deployed at the edge of the network. This means, that Fog nodes can be locally reachable by connected devices nearby, even if the Internet is not available. On the bottom layer, geographically distributed social sensors are connected to their close-by Fog nodes, either directly or by using other social sensors as relays.

As there is a heterogeneity of use cases for Fog computing, there are many different notions of a Fog node. In the following, we provide an overview of current proposals and products (cf. Figure 2). With the advent of computationally stronger network equipment, especially routers, it has been proposed that computations are already performed in the network. For instance, Cisco offers their IOx platforms on hardened routers [6] that are capable of performing data processing tasks. On a higher layer, mini-computers like Raspberry Pi have gained popularity, as they provide acceptable computation performance for a very low price. Additionally, the energy efficiency and miniaturization of those devices allow them to run in environments that were not specifically designed to host computers, i.e., outside of data centers. Mini-computers can even be deployed on drones [9] and provide a completely new level of "mobile computing". A swarm of drones can build an ad-hoc network, a so-called Flying Ad-Hoc Network (FANET) [21], and this
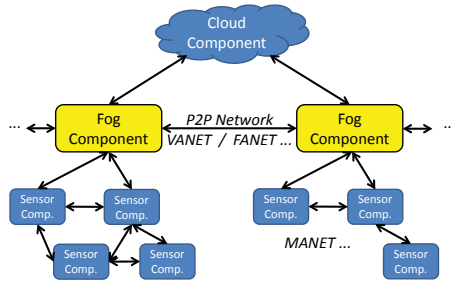
Figure 3: Generic Software Architecture.

way provide Fog computing in an area that lacks any infrastructure. Generally, the deployment of Fog services can be facilitated by using recent lightweight container technology like Docker [4].

The social sensors can be *smart sensors* that perform the sensing, but also filtering and aggregation. In the scenarios described, typically the smart sensors would be connected to smart phones which have certain computational capabilities to do the filtering and aggregation. This reduces the communication overhead between social sensors and Fog nodes, and also reduces computational overhead on the Fog nodes.

2.2 Fog-Enabled Social Sensing Services

Here, we analyze how social sensing services can exploit the Fog infrastructure. They should be able to operate on local information provided on a single Fog node, but also capable of sharing information and collaborating with social sensing services running on neighboring Fog nodes that are reachable. Finally, if the Cloud is reachable, the social sensing services on the different Fog nodes should be able to share global information via the Cloud.

We propose a *generic software architecture* for social sensing applications that is capable of exploiting the Fog infrastructure (cf. Figure 3). It consists of three components: (i) A central management components placed in the Cloud infrastructure (the *Cloud Component*), (ii) A data processing component placed in the Fog infrastructure (the *Fog Component*) and (iii) a social sensing component deployed on the users' devices (the *Sensor Component*). In the following, we detail the tasks of the components and provide an API on all the components for the developers of social sensing services.

2.2.1 Cloud Component. The Cloud Component is, first of all, responsible for the deployment and management of the Social Sensing Service artifacts (program code, meta-data, settings, etc.) on the Fog Components, when an Internet connection is available. This means that the Cloud Component sends software updates to the Fog Components—which can be forwarded to the Sensor Components from there—and also gathers status information from the Fog Components. This can, e.g., help in deciding where to deploy more Fog Components. Furthermore, the Cloud Component can obtain a global view of the sensed data, which can be useful for offline analysis, e.g., in the aftermath of a natural disaster.

2.2.2 Fog Component. The Fog Component is responsible for querying reachable social sensors for data. Such data queries can be formulated in state-of-the-art Stream Processing or Complex Event Processing query languages, such as CQL [3] or Tesla [7].

This allows for defining *continuous queries* that employ windows, aggregation functions and event patterns on the data streams. This way, the Sensor Components can aggregate and filter the sensed data before transferring it to the Fog nodes. Besides traditional continuous queries, queries can also ask for one-time manual sensing, e.g., querying a person to take photos of a specific scene or provide feedback about the number of persons in the vicinity.

The results of the sensing queries are further aggregated in the Fog Component. Data streams from different social sensors need to be correlated so that an overall picture of the situation can be derived [14]. Furthermore, the comparison of information from different sources can improve the information quality [17].

Besides querying and aggregating sensor data from the social sensors in its vicinity, the Fog Component manages the sharing of information between different Fog nodes that are connected to each other over the network. This can be achieved with a peer-to-peer based communication network. Furthermore, if an Internet connection is available at a Fog node, local data can be streamed to the cloud for further analysis. Note that in latency-tolerant applications, mobile Fog nodes can serve as "data mules" that collect sensor data from an area that is disconnected from the Internet, transport that data to an area where the Fog node has Internet connectivity and forward the data to the cloud from there. This can be useful for retrieving sensor data from remote areas.

2.2.3 Sensor Component. The Sensor Component gathers the raw social sensing data, performs the queries from the Fog Component on that data and returns the results back.

It should be noted that not all Sensor Components might be able to directly connect to a Fog Component. This can be due to their physical distance to the next Fog Component, or due to device limitations (e.g., supporting the communication requirements of the Fog Components). For instance, if the Fog Components all require 4G connectivity, some of the Sensor Components might not be able to directly connect to any Fog Components at all. Still, such Sensor Components could connect to other Sensor Components in their proximity, for instance, using WiFi networking. Then, a Sensor Component with direct access to a Fog Component serves as a *communication relay* between the other Sensor Components and the Fog. Such a network can, for instance, be realized with methods from Mobile Ad-Hoc Networks (MANETs). A similar idea was presented by Yusuf, et al [20] with the micrograph middleware. It shows how to handle discovery and manage these distributed and isolated communities for social networks.

Note that as the Sensor Components can be disconnected from the Fog at any time, e.g., because the Fog Component goes down, continuous queries on the Sensor Components should be *soft state*, i.e., employ a time-out mechanism; when the connection to the Fog layer is interrupted for a long time, the sensing is stopped to save energy on the social sensing devices.

2.2.4 Programming API. We propose basic APIs for the Cloud, Fog and Sensor Components (cf. Figure 4). Designers of social sensing services can use those APIs in order to design their system. These APIs are based on a *reliable and delay-tolerant publish-subscribe middleware* [18], which allows information dissemination between nodes in a dynamic network, which is exactly the case with social sensing.

Component	Primitive	Parameters
Cloud	publish	topic, message
	subscribe	topic, message handler
Fog	on_sensor_connection	sensor node
	on_fog_connection	fog node
	on_cloud_connection	cloud node
	query_specific_sensor	sensor node, query
	query_all_sensors	no. of answers, query
	publish	topic, message
	subscribe	topic, message handler
Sensor	on_specific_query	fog node, query
	on_general_query	fog node, query
	publish	topic, message
	subscribe	topic, message handler

Figure 4: Programming API for Social Sensing Services.

API of the Cloud Component.

publish: Cloud component can publish to a specific topic, which serves as the medium of sending message to fog nodes.

subscribe: Cloud Component can subscribe to messages published to the specified topic by a connected Fog Component, and specify a handler function to be called when such a message arrives.

API of the Fog Component.

on_sensor_connection: Called when a sensor connects to the Fog node. The corresponding Sensor Component is registered in the Fog Component, so that future queries can be directed to that Sensor Component. In the registration process, context information of the sensor is exchanged, e.g., the location, and whether other neighboring Sensor Components are reachable by the connecting Sensor Component.

query_specific_sensor: Primitive for querying a specific Sensor Component. The query can be a continuous query formulated in a Stream Processing or a Complex Event Processing query language, but also a one-time query that requires manual action by the social sensor, e.g., taking a photo of a specific scene or reporting how many persons are in the social sensor's vicinity.

query_all_sensors: Primitive for broadcasting a query to sensor components, expecting to get a specific number of query responses, without needing to specify which specific Sensor Components should receive that query. This is useful to achieve a high-fidelity view of the situation by aggregating multiple responses.

publish: The Fog Component can publish a message to a specific topic to convey information to peer fog nodes or the cloud service.

subscribe: The Fog Component can subscribe to messages published to the specified topic, and specify a handler function to be called when such a message arrives.

API of the Sensor Component.

on_specific_query: Handler called when Sensor Component receives a query. If it is a continuous query, it installs the query, processes the sensed data accordingly and publishes results to the given topic. If it is a one-time query that requires manual action by the social sensor, it starts the appropriate routine, e.g., a message on the display of a smart phone.
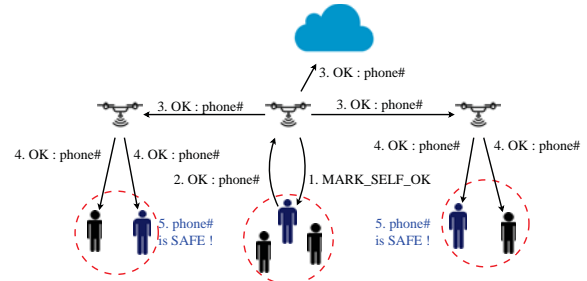

Figure 5: Schematic of Family Safety application (members of a family are distinguished by using blue color).

on_general_query: The Sensor Component receives a query that requires a given number of answers. The Sensor Component uses broadcast or multicast protocols, e.g., gossiping, to disseminate the query to an appropriate number of other Sensor Components.

publish: Sensor components can publish to a specific topic, which serves as the medium of returning query results to fog nodes.

subscribe: Sensor Component can subscribe to messages published to the specified topic by a connected Fog Component, and specify a handler function to be called when such a message arrives.

To create a Fog-enabled social sensing service, the Cloud Component, Fog Component and Sensor Component can implement the proposed API. In the next section, we give concrete examples of how the proposed API can be used in practice.

3 CASE STUDIES

In this section, we discuss a couple of use cases to illustrate how social sensing services can be adapted to the proposed Fog infrastructure. We show that existing services could benefit from the deployment in the Fog by providing local services to their users in the face of Internet connectivity problems.

3.1 Social Sensing Services in Natural Disasters

Social sensing services deployed on the Fog can help to gather and disseminate local knowledge among the affected people. Owing to the relatively local nature of the information pertaining to a disaster-prone area, Fog Computing is destined for providing the required connectivity to affected people and emergency response teams so that they can help mitigate the adverse effects.

3.1.1 Checking the Safety Status of Family Members. Recall the use case that a person wants to check whether his family members in a disaster-stricken area are OK. We have discussed that the Internet connectivity can be interrupted, so that his family members cannot access cloud services like Facebook Safety Check[2]. In such a situation, a Fog-enabled Safety Check Service can enable disaster-struck people to ascertain the safety of their family members, as shown in Figure 5.

A user interested in knowing whether his family members are okay starts the Sensor Component of the safety check service on his smartphone. The Sensor Component (SC) collects the list of phone numbers belonging to the user's family members.

Whenever a Fog Component (FC) is able to connect to a SC, it queries that SC with a specific query asking to mark itself OK. Upon

---

[2]https://www.facebook.com/about/safetycheck/

receiving this query, the SC displays an alert on the smartphone's screen asking the user to mark himself OK. Then, the SC publishes the user's phone number to its local FC, so that the FC can know about this user's safe situation.

All FCs subscribe to the topic named *OK*, to which both SCs and peer FCs publish. The pub/sub system is the medium of disseminating information of safe users over the area. When an FC receives a new *safe* phone number, it publishes it on the topic "OK" so that it may be received by SCs and peer FCs. SCs also subscribe to the topic *OK* so that they receive information about the people who are safe. If an SC receives a *safe* phone number and it belongs to one of the user's family members, she is alerted of their safety.

Algorithm 1 Fog component: Family Safety application

---

procedure INIT
    subscribe(OK, *on_ok_callback*)
    safeList ← {}   ▷ People this node knows are OK
end procedure
procedure ON_OK_CALLBACK(p)
    if $p \notin safeList$ **then**
        publish(OK, p)   ▷ publish phone# to OK people topic
        $safeList \leftarrow safeList \cup \{p\}$
    end if
end procedure
procedure on_sensor_connection(*sensor*)
    $query\_specific\_sensor(sensor, MARK\_SELF\_OK)$
end procedure

Algorithm 2 Sensor component: Family Safety application

---

procedure INIT
    subscribe(OK, *on_ok_callback*)
    family ← GetFamily()
end procedure
procedure ON_OK_CALLBACK(*phone#*)
    if $phone\# \in family$ **then**   ▷ if recv. phone# is family
        DISPLAY(f is SAFE)
    end if
end procedure
procedure on_specific_query(*type*)
    if $type == MARK\_SELF\_OK$ **then**
        DISPLAY(Mark yourself OK)
        if user marked OK **then**   ▷ if user marks himself OK
            publish($OK, phone\#$)
        end if
    end if
end procedure

3.1.2 Population Density Map for Emergency Response Teams. Suppose that an emergency response team wants to get information about the distribution of individuals in an area struck by a hurricane. To this end, it needs aggregated information about the sectors in which the persons are located. Based on this, the emergency response deduces where to go first to help or evacuate people.

The emergency response team installs a continuous query on the Fog Component of the area they plan to go to, querying for detailed information of how many Sensor Components are connected in which area. The continuous query is installed on the Fog Component using the primitive query_all_sensors(∞, pos_query). As a result, all connected Sensor Components report their position. This data is used for building a density map of persons in specific sectors, which is returned to the emergency response team.

3.2 Social Sensing Services in Economically Under-Served Regions

3.2.1 Modernizing Public Transportation. Using Fog Computing, social sensing services can learn about the transportation infrastructure: density of users in a bus, the variability of the service, predicted time tables, and other metrics. When facing intermittent Internet connectivity, Fog nodes can provide local connectivity to the users. To this end, each bus carries a small Fog computing device. Passengers in the bus can connect with their smart phones as social sensors, so that the Fog Component on the bus receives sensor data from the smart phones. For instance, the Fog Component can query the destination of the passengers (using the query_all_sensors(∞, destination_query) primitive), so that an optimal bus route is calculated.

Buses can share information with each other when they are close enough to build a vehicular ad-hoc network (VANET) [12]. To this end, they use the publish primitive. Reliable and delay-tolerant publish/subscribe ensures that buses can also share information with the cloud when an Internet connection is available. The shared information can concern, e.g., road and traffic conditions, so that the bus driver can benefit from the knowledge available in other buses as well. Passengers in the bus can connect to the Fog Component to receive information on the expected arrival time at their destination, by subscribing to the appropriate topic.

If passengers do not have smart phones that can carry Sensor Components of the system, installed Sensor Components on-site can provide them a basic service. For instance, a "call a bus" button can be installed at a bus stop. If a passenger needs a bus ride, she can press the button; the social sensing service will contact buses that are close-by via the Fog infrastructure and adapt the route such that the next bus will visit the queried bus stop. Such simple sensors are cheap enough to be deployed in rural and economically under-served areas in a large number. Additionally, the information obtained can be used by local governments to improve the service and to ask other authorities for services that adapt better to the communities that are being served.

3.2.2 Sensing the Status of Infrastructure. In remote regions, it is difficult to monitor installed infrastructure, such as street lighting and solar panels, for failures. Employing them with sensors that sense the infrastructure status promises to facilitate the monitoring. Such sensors are available for a very low price. However, in rural areas, there might not be Internet connectivity to read the sensed data remotely. Even if Internet connectivity is available, providing the sensors with an Internet connection increases their price, for acquisition as well as operation.

To this end, the idea of "data mules" has been proposed. People or moving objects such as buses can serve as a data mule to collect sensor data and upload it to the Cloud as soon as Internet connectivity is available. This has been proposed in order to deliver

emails to remote regions [16]. However, it is not directly applicable to sensor data, which might have much larger scales. When there is a lot of data to be sensed, the storage of micro-computers could be over-utilized. Furthermore, the Internet connection, when available, could still have a very low bandwidth, so that the upload of Gigabytes of data is infeasible. Extending the simple data mule concept to a mobile Sensor Component can help to handle this issue. The Sensor Components execute aggregation and filtering operations, so that the amount of data transmitted is much lower.

4 TECHNICAL CHALLENGES

The Fog infrastructure poses a large range of technical challenges on the implementation. For example, if the Fog nodes are installed on drones, different communication protocols are used and coupling between them is required. Additionally, the network protocols need to be latency-tolerant; each node needs to be able to queue messages until a connection is reestablished.

Handling geo-distributed resources is challenging. Part of the complexity is defining the type of algorithm to deploy on the nodes based on the available capabilities. This has to be added to the process of deploying applications to nodes with limited Internet connectivity and untrusted infrastructure.

Common distributed systems issues also arise in the context of Fog social sensing. Fog resources might have lower availability and dependability than servers in cloud data-centers. One of the main challenges is that protocols and middleware need to be distributed and energy-efficient, e.g., discovering other peers and fog nodes without a central entity and with limited energy. Load balancing is another common issue. For example, the region of a disaster may require more resources such as networking and computing. How can the Fog infrastructure be organized to meet different resource demands? Mobility of Fog nodes could be used to dynamically balance the pressure on each Fog node.

There also exist social sensing specific challenges. Social sensing can bring disinformation and inaccuracy [17]. The identity of the users and correctness of the information cannot be guaranteed [15]. These errors are not necessarily intended, but caused by mistakes and misunderstandings. Fog computing itself enhances the sharing of information within the region responsible for a given Fog node. However the question arises, is there more we can do to provide reliable information sharing? For example, an intuitive idea is to gather the information from different social sensors and eliminate outliers. A further question is how to route the information to the intended receivers. Simple flooding will lead to each user receiving too much information and bringing pressure to the network infrastructure. When the Fog nodes are mobile, this issue becomes more challenging due to non-deterministic connectivity.

5 CONCLUSION AND OUTLOOK

In this paper, we have extended the vision of Fog Computing toward providing social sensing services in situations when Internet connectivity is limited. We have outlined the basic design principles of such a Fog-enabled social sensing service, and have proposed a generic API that social sensing services can employ in order to use the Fog infrastructure.

A Fog-based social sensing platform could help in the transition of existing cloud-based social sensing services to the Fog. Such a middleware needs to provide basic building blocks for the developers of a social sensing service, and also include mechanisms for deployment, communication management, and many other technical details. It is imaginable that cloud-based services could be transformed toward the Fog platform in a semi-automatic manner. This way, many of the existing useful cloud-based services could benefit from the emerging Fog infrastructure.

REFERENCES

[1] Charu C Aggarwal and Tarek Abdelzaher. 2013. Social sensing. In *Managing and mining sensor data*. Springer, 237–297.
[2] Maura C Allaire. 2016. Disaster loss and social media: Can online information increase flood resilience? *Water Resources Research* 52, 9 (2016), 7408–7423.
[3] Arvind Arasu, Shivnath Babu, and Jennifer Widom. 2006. The CQL Continuous Query Language: Semantic Foundations and Query Execution. *The VLDB Journal* 15, 2 (June 2006), 121–142.
[4] Paolo Bellavista and Alessandro Zanni. 2017. Feasibility of Fog Computing Deployment Based on Docker Containerization over RaspberryPi. In *Proceedings of the 18th International Conference on Distributed Computing and Networking (ICDCN '17)*. ACM, Article 16, 10 pages.
[5] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. 2014. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 169–186.
[6] Cisco. 2017. Cisco DevNet: IOx. https://developer.cisco.com/site/iox/. (2017). [Online; accessed 19-January-2017].
[7] Gianpaolo Cugola and Alessandro Margara. 2010. TESLA: A Formally Defined Event Specification Language. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems (DEBS '10)*. ACM, 50–61.
[8] Dirk Eilander, Patricia Trambauer, Jurjen Wagemaker, and Arnejan van Loenen. 2016. Harvesting social media for generation of near real-time flood maps. *Procedia Engineering* 154 (2016), 176–183.
[9] Y. Gu, M. Zhou, S. Fu, and Y. Wan. 2015. Airborne WiFi networks through directional antennae: An experimental study. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)*. 1314–1319.
[10] Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. 2013. 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 (MCC '13)*. ACM, 15–20.
[11] Yu Liu, Xi Liu, Song Gao, Li Gong, Chaogui Kang, Ye Zhi, Guanghua Chi, and Li Shi. 2015. Social Sensing: A New Approach to Understanding Our Socioeconomic Environments. *Annals of the Association of American Geographers* 105, 3 (2015), 512–530.
[12] Christian Lochert, Hannes Hartenstein, Jing Tian, Holger Fussler, Dagmar Hermann, and Martin Mauve. 2003. A routing strategy for vehicular ad hoc networks in city environments. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*. IEEE, 156–161.
[13] B.S. Manoj and Alexandra Hubenko Baker. 2007. Communication Challenges in Emergency Response. *Commun. ACM* 50, 3 (March 2007), 51–53.
[14] Ruben Mayer, Boris Koldehofe, and Kurt Rothermel. 2015. Predictable Low-Latency Event Detection With Parallel Complex Event Processing. *IEEE Internet of Things Journal* 2, 4 (Aug 2015), 274–286.
[15] Raina M Merchant, Stacy Elmer, and Nicole Lurie. 2011. Integrating social media into emergency-preparedness efforts. *New England Journal of Medicine* 365, 4 (2011), 289–291.
[16] A. Pentland, R. Fletcher, and A. Hasson. 2004. DakNet: rethinking connectivity in developing nations. *Computer* 37, 1 (Jan 2004), 78–83.
[17] Tomer Simon, Avishay Goldberg, and Bruria Adini. 2015. Socializing in emergencies - A review of the use of social media in emergency situations. *International Journal of Information Management* 35, 5 (2015), 609 – 619.
[18] Magnus Skjegstad, Frank T Johnsen, Trude H Bloebaum, and Torleiv Maseng. 2012. Mist: A reliable and delay-tolerant publish/subscribe solution for dynamic networks. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*. IEEE, 1–8.
[19] R. Stiegler, S. Tilley, and T. Parveen. 2011. Finding family and friends in the aftermath of a disaster using federated queries on social networks and websites. In *2011 13th IEEE Int'l Symposium on Web Systems Evolution (WSE)*. 21–26.
[20] L. Yusuf and U. Ramachandran. 2012. Community Membership Management for Transient Social Networks. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*. 1–7.
[21] lker Bekmezci, Ozgur Koray Sahingoz, and amil Temel. 2013. Flying Ad-Hoc Networks (FANETs): A survey. *Ad Hoc Networks* 11, 3 (2013), 1254 – 1270.