On-demand Self-adaptive Data Analytics in Large-scale Decentralized Networks

Evangelos Pournaras and Jovan Nikolić
Professorship of Computational Social Science
ETH Zurich
Zurich, Switzerland
{epournaras,jnikolic}@ethz.ch

***Abstract*—The Internet of Things empowers citizens to interconnect their devices, such as smart phones, into large-scale participatory decentralized networks, which they can use to make real-time collective measurements as public good, for instance, crowd-sourcing the monitoring of traffic in a city. This approach is an alternative to big data analytics systems that are often expensive to access, privacy-intrusive and allow discriminatory and profiling actions over citizens' data. On the contrary, large-scale decentralized networks are complex to manage and collective measurements, i.e. computations of aggregation functions, need to encounter several dynamics such as continuously changing input data streams and highly varying temporal demand for access to the collective measurements. This paper proposes a highly reactive self-adaptation model to tackle the challenge of dynamic computational demand in large-scale decentralized in-network aggregation. The self-adaptation process makes nodes self-aware about other nodes that join and leave the network and therefore it makes them capable of self-orchestrating the communication to improve accuracy and minimize communication cost. The model is simple, yet agile. This is shown when applied in DIAS, the Dynamic Intelligent Aggregation Service without introducing architectural changes. Evaluation using data from a real-world smart grid pilot project as well as extreme demand profiles that scale up and down the demand 50% on average confirm the cost-effectiveness of in-network aggregation empowered by self-adaptation. The findings are confirmed both in simulation and a large-scale live deployment in a cluster infrastructure with 3000 independent Java virtual machines each running a DIAS node. Overall, the results encourage new promising pathways towards the broader adoption of self-adaptive participatory data analytics in large-scale decentralized networks.**

***Keywords*-self-adaptation; aggregation; accuracy; data analytics; decentralized network; participation; agent, gossip communication**

I. Introduction

Large-scale decentralized networks running the Internet of Things can be formed in a bottom-up way by citizens to perform real-time participatory data analytics as public good, in contrast to centralized big data analytics that are usually privacy-intrusive, and may allow discriminatory and profiling actions over citizens' data [1], [2]. This is especially the case in non-critical applications that can tolerate inaccuracies and high network dynamics, for instance, citizens who self-organize and employ decentralized data analytics for a higher community self-awareness about safety, traffic, social networks, human mobility, energy consumption and other [3], [4], [5], [6], [7]. However, large-scale decentralized networks are highly complex to control and manage, especially when distributed computations need to adapt to dynamics such as varying demand by citizens, who may choose to autonomously participate or not in the network, with implications to the quality of service, i.e. accuracy, and communication cost experienced in the network. This paper studies self-adaptation as the means to cost-effectiveness in such dynamic environments, and therefore to the feasibility of decentralized data analytics. The focus of this paper is the design of a simple, yet agile self-adaptation process to dynamic computational demand that can make large-scale decentralized networks capable of performing accurate data analytics without unnecessarily consuming network resources under perturbations in computational demand.

978-1-5386-1465-5/17/$31.00 ©2017 IEEE

A dynamic computational demand occurs when users temporarily request access to the results of real-time data analytics operations, for instance, the total load of the power grid [8], or the average traffic in a city [9]. The nodes of a decentralized network that performs in-network aggregation need to communicate and exchange their values so that the estimations of aggregation functions, i.e. SUMMATION and AVERAGE, converge to the actual 'true' values. When the demand scales up and down, meaning new nodes join and request access to the aggregates, the accuracy drops dramatically and communication between nodes needs to increase to encounter for the new demand, i.e. request access to the output of the aggregation functions. There is a plethora of work on how to perform intelligent resource allocation and provision in centralized computational systems such as cloud computing and big data infrastructures [10], [11], [12]. However, there is very limited work that studies the effect of varying computational demand on the cost-effectiveness of decentralized in-network aggregation, which actually turns out to be a challenge of efficient communication and distributed data management rather than a parallel computational problem.

This paper introduces a novel model of self-adaptation to dynamic computational demand. The model is based on gossiping communication that allows each node in a network to become self-aware about new nodes joining the network. By detecting the joined nodes, communication is reactively self-orchestrated so that the shared data reach the nodes that join the network. As a result, the accuracy of aggregation rapidly increases. On the contrary, when nodes leave the network, the

communication automatically diminishes and therefore network resources are used more efficiently. The model is applied to DIAS, the *Dynamic Intelligent Aggregation Service* and is evaluated under several extreme demand profiles in which on average 50% of the nodes join and leave the network. Data from a real-world smart grid pilot project are fed in a network operating in simulation as well as in a live network deployment with 3000 Java virtual machines communicating in parallel over a cluster network. Results confirm the feasibility of aggregation under dynamic computational demand. Trade-offs between accuracy and communication cost are illustrated.

In summary, the contributions of this paper are the following:

- A self-adaptation model for large-scale decentralized data analytics under dynamic computational demand.
- The applicability and evaluation of the self-adaptation model in DIAS, the Dynamic Intelligent Aggregation Service. The implemented model is a new system feature that makes DIAS self-adaptive to a varying number of aggregators in the network.
- Measurements that compare and contrast simulation and the live network deployment of DIAS operating with 3000 parallel Java virtual machines under highly dynamic computational demand.

This paper is organized as follows: Section II introduces the model of self-adaptive data analytics to dynamic computational demand. Section III illustrates the applicability of the proposed model in DIAS, the Dynamic Intelligent Aggregation Service. Section IV experimentally evaluates the proposed model under extreme synthetic demand profiles in simulation and live deployment using real-world smart grid data. Section V compares the proposed self-adaptation model with related work. Section VI concludes this paper and outlines future work.

II. A Self-adaptive Model for Dynamic Computational Demand

The proposed model is designed for data analytics over large-scale decentralized networks in which *data suppliers* share data and *data consumers* collect aggregate information computed over the shared data. This paper focuses on lightweight data analytics such as the in-network computation of aggregation functions e.g. SUMMATION, AVERAGE, MAXIMUM, STANDARD DEVIATION, TOP-K, etc, which are challenging to accurately estimate in a decentralized network [13], [14]. *In-network aggregation* makes available to every node participating in the network the output of the computed aggregation functions using as input the values of all nodes connected to the network.

A user running an application on a smart phone or a personal computer connects to a node of a decentralized network with computational resources allocated for in-network aggregation. These resources can be the users' devices, a collection of distributed servers for this purpose [15], [3] such as the community pods in the Diaspora social network, or a cloud computing infrastructure [16]. The user connects to the *network node* as data supplier, data consumer or both. The node contains two *agents*: (i) the *disseminator* and (ii) the *aggregator*. A data supplier shares its data with the disseminator agent that is responsible to remotely communicate with other online aggregators in the network and provide them with the shared data. A data consumer acquires access to the output of the aggregation functions via the aggregator agent in the node. The aggregator collects the shared data of disseminators and computes aggregation functions, whose output is made available to the data consumers. Depending on the system setup, use-case and application domain, access to aggregates can be free, or it can also be an on-demand 'pay as you go' service such as the pricing models that cloud computing infrastructures adopt [17].

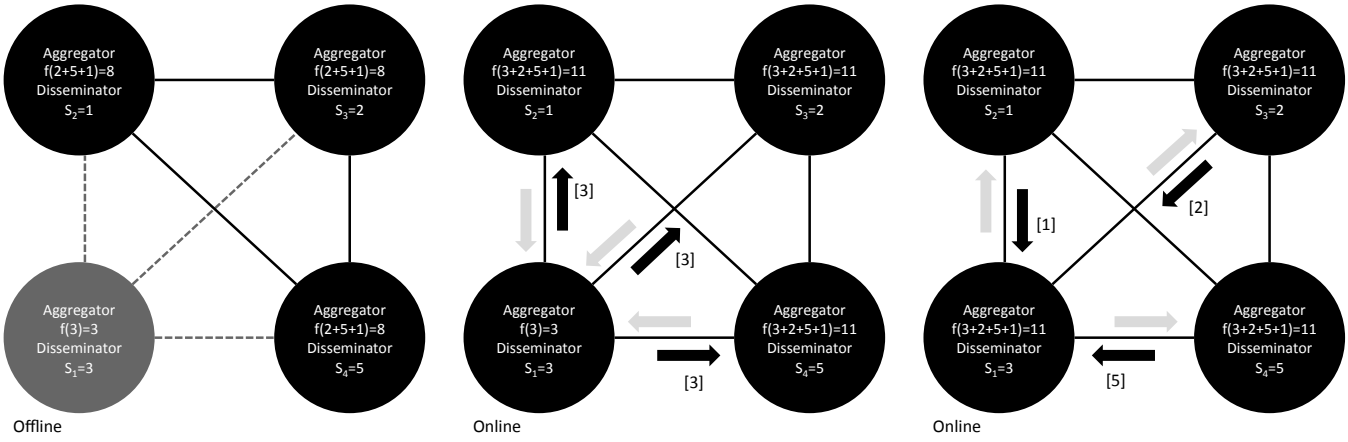
The proposed model focuses on the design of a self-adaptation process to dynamic computational demand. A dynamic computational demand stems from the autonomy of data consumers to participate on-demand in the network. An automated self-adaptation is required to continuously scale up or down the performed computations to only the nodes participating with a data consumer. Figure 1 illustrates the concept of in-network aggregation when a node joins the network.

The self-adaptation process concerns the fully decentralized detection of online and offline aggregators. Disseminators monitor the network and discover aggregators that either (i) recently joined the network and acquire the shared data of data suppliers or (ii) have outdated information and acquire the most recent shared data of data suppliers.

A decentralized network discovery of aggregators can be achieved with the *peer sampling service* that is a gossip-based communication mechanism [18]. Gossiping nodes maintain a list of randomly selected *node descriptors*, the *view*, which has a fixed size and is periodically exchanged with other nodes via peer-to-peer interactions. The node descriptor contains information such as the IP address and port number of the node as well as application information registered for spreading in the network, for instance the *age*[1] of the node. At each gossip exchange between two nodes, the views of the nodes are exchanged. The descriptors that fill the updated view are selected based on two parameters, the SWAPPER and HEALER. SWAPPER selects highly random descriptors, whereas HEALER selects the most recent descriptors, i.e. descriptors with a low age. The peer sampling service provides the following capabilities: (i) A well-connected and robust overlay network that is continuously updated. (ii) A rapid spread of information in the network.

Discovery is achieved as follows: When a data consumer participates in the network, the respective aggregator comes online and connects to the network. It adds a unique identifier (ID) in the local node descriptor of the peer sampling service and periodically resets the age field back to zero. The node descriptor is spread to other remote nodes via the gossip-based

[1]This is a counter that the nodes in the network increment at every periodic execution of the gossiping algorithm. The counter is reset back to zero by the node that creates its descriptor as a way to signal to other nodes that is online.



(a) One offline node.

(b) All nodes are online. The disseminator turned online shares its value.

(c) The aggregator turned online receives the values of disseminators.

Figure 1. In-network aggregation (SUMMATION) and adaptation of computations when a node comes online in the network. At first, one offline and three online nodes are present. Then all nodes come online. The joining disseminator sends its value to the existings nodes in the network. Computations of the aggregation functions are adapted. The other disseminators send their value to the new online node. Aggregation function is accurately computed.

interactions and as a result disseminators have local access to contact information of remote aggregators. By using this information, disseminators establish an *aggregation session* with each aggregator discovered via this process. In this way, the shared data of the suppliers become input to remotely computed aggregations functions, whose output is made available to data consumers. Nevertheless, a data consumer may not anymore participate in the network. In this case, the aggregator goes offline and, as a result, it removes its ID from the node descriptor. Therefore, disseminators do not detect the aggregator over time and no other aggregation sessions are initiated. Algorithm 1 illustrates a high-level algorithm description of the tasks executed by the aggregator.

Algorithm 1 The aggregator tasks orchestrated by the participation of the data consumer in the network. Timers denote the periodic execution of the peer sampling service and aggregators.

```
Require: status from data consumer
loop
  if status is 'connect' then
    set ID in node descriptor
    if peer sampling service timer expired then
      reset age in node descriptor to zero
    end if
    if aggregator timer expired then
      deliver aggregates to data consumer
    end if
  else
    // status is 'disconnect'
    set ID in node descriptor to null
  end if
end loop
```

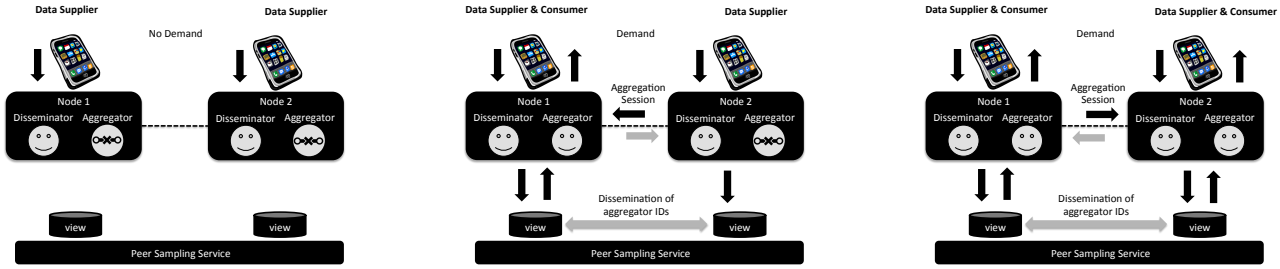
Figure 2 highlights the proposed self-adaptation model for in-network aggregation networks with varying computational demand. Figure 2a shows a network without computational demand as there are no data consumers participating. There are no online aggregators and therefore disseminators do not find in the view IDs of aggregators to communicate with. Figure 2b shows a data consumer coming online in Node 1. It adds its ID to the local node descriptor that is spread to the network via the peer sampling service. The disseminator in Node 2 discovers the remote aggregator and performs an aggregation session. The same process repeats in Figure 2c for Node 2. In this case, each node is connected to a data supplier and consumer.

Note that the self-adaptation process regulates the communication performed in an automated fashion given the computational demand in the network. Self-adaptation is orchestrated by the inner connected nodes of the network. No third parties are introduced for this purpose and therefore the model is highly applicable for large-scale decentralized networked systems.

III. Model Applicability

The proposed self-adaptation model is applied to the decentralized in-network aggregation system of DIAS, the *Dynamic Intelligent Aggregation Service*. This section provides an outline of DIAS, however, a detailed illustration is out of the scope of this paper and it is covered in earlier work [19], [14], [20]. Therefore, the focus of this section is to illustrate how the proposed self-adaptation model is applied to DIAS.

DIAS has *three special features* that distinguish it from other related decentralized aggregation systems [13], [21], [22], [23]: (i) It can compute multiple aggregation functions without changing the executed algorithm [19], [14]. (ii) Estimations of the aggregates can be continuously adapted to encounter changing input data (streams) from each node [14]. (iii) Estimation of the aggregates can be adapted when data



(a) Network with only data suppliers.

(b) A data consumer comes online in the network.

(c) The network consists of both data suppliers and consumers.

Figure 2. A self-adaptive network model to varying computational demand. Data suppliers share data with data consumers. When only data suppliers participate in the network, there is no communication exchange in the network. Aggregators are offline. When data consumers join, aggregators connect and add their IDs in their node descriptor that is spread via the peer sampling service. Disseminators discover these IDs and initiate aggregation sessions with remote aggregators to share the data of their data supplier.

suppliers join and leave the aggregation network or even fail [20]. However, a self-adaptive aggregation when data consumers join and leave the network has not been earlier studied within the context of DIAS as well as in the broader context of decentralized in-network aggregation. This paper bridges this gap and contributes a new feature to DIAS: on-demand self-adaptation to dynamic computational demand, i.e. varying number of data consumers.

The first feature is engineered via aggregation sessions that make locally available to every data consumer the input values of an aggregation function. This is achieved via the discovery of data consumers by data suppliers using the peer sampling service. The second feature is feasible by using the local data model of *possible states*. According to the model, the data supplier does not have to share the raw data it generates as these data may change so rapidly that the changes become irrelevant if they have to be communicated in real-time over a large-scale decentralized network. Moreover, privacy concerns may restrict the sharing of the data in the network [24]. Instead, a data supplier connected to a node i can abstract the time series data within a time window to a finite number k of real values $\{p_{i,1}, ..., p_{i,k}\}, p_{i,j} \in \mathbb{R}, \forall j \in \{1, ..., k\}$, the possible states. For instance, the smart meter power consumption data of an energy consumer can be abstracted to three representative power consumption values representing the low, medium and high energy profiles of the energy consumer [8]. At every time point, each data supplier selects one and only one possible state that is the *selected state* $s_i = p_{i,s} \in \{p_{i,1}, ..., p_{i,k}\}$ of node i. The third feature is based on self-corrective computations performed by migrating agents, whose parent node leaves or fails. Their goal is to reverse (rollback) earlier computations performed as long as their parent node remains disconnected.

The proposed model does not require any major change in the aforementioned design of DIAS. The peer sampling service is reused as part of the existing DIAS design. The communication protocol of the aggregation sessions does not also require any alteration. Therefore, the design effort focuses on interfacing the self-adaptation model with the distributed memory system of DIAS, which relies on probabilistic data structures, the *bloom filters* [25]. The memory system is used by disseminators to select aggregators, among the ones discovered via the peer sampling service, with which aggregation sessions are established. The memory system of a disseminator enables the selection of aggregators that either (i) aggregate for first time a selected state of the disseminator or (ii) have an outdated selected state from the disseminator and require the most recent one. The bloom filters allow for any remote pair of a disseminator and an aggregator to mutually reason about the following nested scenarios:

Reasoning 1. *A disseminator recalls whether an aggregation session has been earlier performed with a certain aggregator. Mutually, an aggregator recalls whether an aggregation session has been earlier performed with a certain disseminator.*

The disseminator and the aggregator perform a local query to a simple bloom filter that stores aggregator and disseminator IDs respectively to implement the Reasoning 1.

Reasoning 2. *A disseminator recalls which possible state an aggregator has earlier aggregated. Mutually, an aggregator recalls which possible state of a certain disseminator has been earlier aggregated.*

In the disseminator side, Reasoning 2 is implemented by a counting[2] bloom filter for each possible state. These bloom filters store aggregator IDs. In the aggregator side, Reasoning 2 is implemented by a counting[2] bloom filter storing IDs of selected states.

The performance of the DIAS aggregation when aggregators come online or go offline can be measured with the following two metrics:

- *Accuracy*: This is the average relative error of the aggregates computed by the online aggregators and it is defined by the absolute difference between actual and estimated values divided by the actual values. The accuracy evaluates the effectiveness of the DIAS network to perform good estimations of the aggregations functions.

[2]A counting bloom filter allows removal of elements, which is required when data suppliers change selected states so that they distinguish whether aggregators have aggregated the most recent selected state.

- *Communication cost*: The total number of DIAS messages exchanged between disseminators and aggregators. The communication cost of DIAS originates from two messages for every aggregation session performed. The communication cost of the peer sampling service is out of the scope of this paper as it is constant and relies entirely on the execution period [18].

If a data consumer participates in the network, its aggregator is online. Disseminators can detect it via the peer sampling service and the aggregator ID present in the received node descriptor. An aggregation session is initiated for every change of a selected state performed by the disseminator. In this way, a high accuracy is preserved.

If a data consumer does not anymore participate in the network, the aggregator removes its ID from the node descriptor. Over time, the earlier node descriptors containing the aggregator ID vanish and the latest node descriptors do not have an aggregator ID. Therefore, no aggregation sessions are performed. The communication cost is minimized, while accuracy is not influenced by the aggregators going offline, given that accuracy is measured by the online aggregators.

IV. Experimental Evaluation

The peer sampling service[3] and the DIAS aggregation service are implemented in Java using the Protopeer distributed prototyping toolkit [26]. They are deployed in the Euler[4] high performance cluster infrastructure of ETH Zurich. Experimentation is performed in both *operational modes* of Protopeer: (i) *simulation* and (ii) *live*. In live mode, every node of the network runs its own Java virtual machine and communicates by exchanging network messages using the Apache MINA library[5]. The same DIAS implementation is used for both operational modes, however, the live mode requires several complex scripts to bootstrap and synchronize the network deployment given various computational constraints that cluster infrastructures have that are out of the scope of this paper. Results are mainly illustrated for the simulation mode as they are free of deployment artifacts. In contrast, the bootstrapping process of a live experiment requires that all nodes are online and can communicate with the bootstrap server under infrastructural and administration constraints such as network delays, varying computational loading, synchronization issues and others. A performance comparison between simulation and live is illustrated in this section.

DIAS is fed with real-world data[6] from a state of the art pilot project about the electricity consumption in Ireland: the *Electricity Customer Behavior Trial* (ECBT). The project ran during the period 2009-2010 with 6435 residential and small-medium enterprise consumers, from which 3000 residential consumers are used in the illustrated experiments to decrease the execution time of the experiments. Power consumption data are collected from smart meters every 30 minutes. Data from date 4.1.2009 are used for the experiments. The total records of raw data used in the experiments are 2 records/hour*24 hours=48 records. Possible states are extracted from the raw data by performing clustering with k-means, where $k = 5$, using the Weka library[7]. The motivation behind using real-world energy consumption data is mainly the validation with realistic user data. Yet, in the context of demand-side energy management, DIAS is applicable when participatory citizens self-organize in a bottom-up way to support trusted cooperatives and collaborative schemes for monitoring their collective demand and optimizing their energy usage [27], [28], [29], [30].

The epoch duration[8] is selected to be 1 sec (1/4=0.25 sec for the peer sampling service). A high execution rate of DIAS improves convergence speed but also increases the communication rate, which is though minimized to zero after convergence. Out of the total 800 epochs, the first 100 epochs are used for system bootstrapping. Aggregation is performed in the next 14*48=672 epochs. The view size of the peer sampling service is 50 with a SWAPPER parameter of 24 and a HEALER parameter of 1 [22]. Each of the 3000 nodes of DIAS is equipped with a disseminator to test the most demanding scenario. A maximum number of 40 aggregation sessions per epoch are initiated by each disseminator. Results for the AVERAGE, SUMMATION and MAXIMUM aggregation functions are presented.

To stretch the performance of the aggregation performed, two extreme synthetic demand profiles are introduced: (i) *UP-DOWN* and (ii) *DOWN-UP*. Figure 3a and 3b illustrate the demand profiles used for the experimental evaluation of this paper. Both profiles have out of the maximum of 3000 aggregators 1500 online (50%) on average over the 800 epochs. All potential changes in the network are scheduled to start on the 100th epoch and finish on the 739th epoch. In the UP-DOWN profile, the number of aggregators is minimum on the 100th and 739th epoch and maximum on the 419th epoch. In contrast, in the DOWN-UP phase the number of aggregators is minimum on the 419th epoch and maximum on the 100th and 739th epoch. Three *scaling strategies* (SS) are evaluated in which a different number of aggregators is added or removed periodically at different *scaling steps* such that the average number of aggregators during runtime is 1500: (i) SS-1 with period 160 and 1600 nodes per scaling step, (ii) SS-2 with period 80 and 800 nodes per scaling step and (iii) SS-4 with period 40 and 400 nodes per scaling step.

Figure 3c and 3d show how the UP-DOWN and DOWN-UP demand profiles appear in the live operational mode. Given that the system is designed to operate in a fully asynchronous fashion with each node having its own clock, the demand profiles are not anymore step functions, though

[3]Available at https://github.com/epournaras/PeerSamplingService (last accessed: September 2017

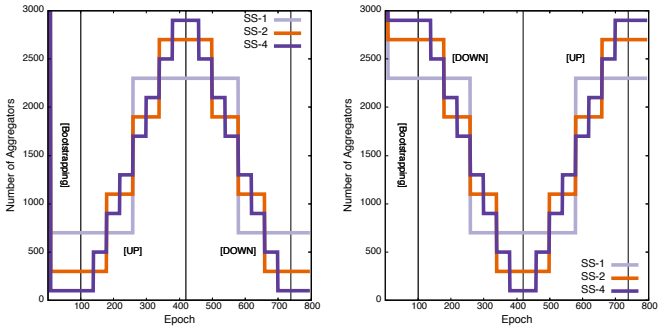
[4]Available at http://brutuswiki.ethz.ch (last accessed: September 2017).

[5]Available at https://mina.apache.org (last accessed: September 2017)

[6]Available at http://www.ucd.ie/issda/data/commissionforenergyregulationcer/ (last accessed: September 2017)

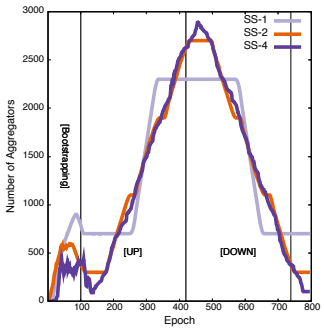
[7]Available at https://weka.wikispaces.com (last accessed: September 2017)

[8]The epoch duration would be 30 minutes/14 DIAS executions=2.14 minutes (2.14/4=0.5 minutes for the peer sampling service) if DIAS operated for ECBT.

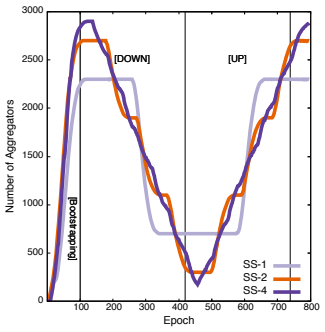


(a) Simulation, UP-DOWN

(b) Simulation, DOWN-UP

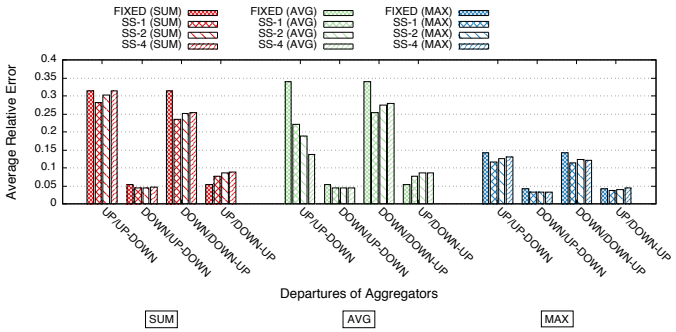


(c) Live, UP-DOWN

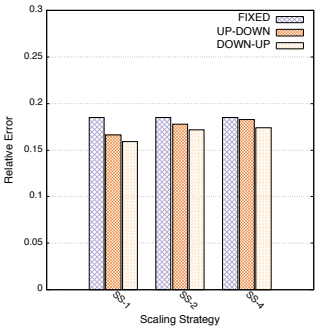


(d) Live, DOWN-UP

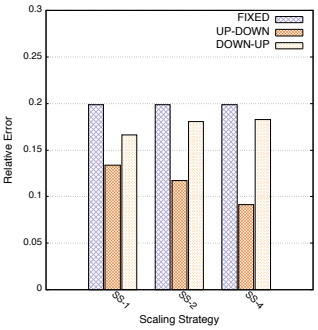
Figure 3. Number of aggregators over runtime for the three scaling strategies.



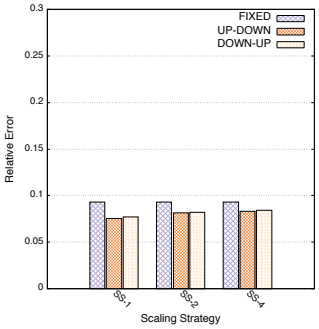
(a) Overview



(b) SUMMATION



(c) AVERAGE



(d) MAXIMUM

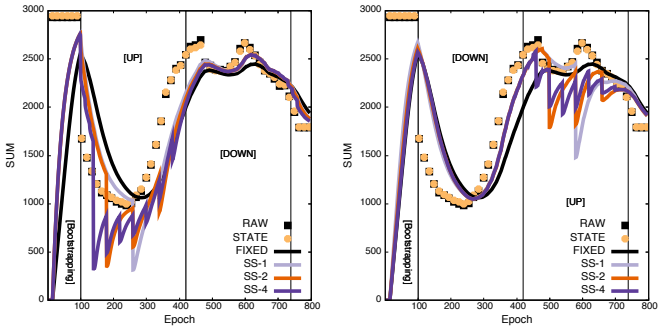
Figure 4. Accuracy of the three aggregation functions for each scaling strategy and demand profile.

they approximate the original designed demand profiles of Figure 3a and 3b.

Figure 4a illustrates in detail the accuracy of the aggregation functions for each scaling strategy and demand profile, whereas, Figure 4b-4d show a more summarizing view of the accuracy. The following observations can be made: (i) The average relative error of SUMMATION under UP-DOWN is 34.95% and 54.43% higher than AVERAGE and MAXIMUM respectively. However, the average relative error of AVERAGE under DOWN-UP is 4.58% and 54.04% higher than SUMMATION and MAXIMUM. (ii) In SUMMATION under UP-DOWN, the average relative error for SS-1 is 6.64% and 9.67% lower than SS-2 and SS-4 respectively. For DOWN-UP, SS-1 is 8.10% and 9.45% lower than SS-2 and SS-4 respectively. In AVERAGE under UP-DOWN, the average relative error for SS-1 is 12.29% and 31.77% higher than SS-2 and SS-4 respectively. For DOWN-UP, SS-1 is 8.64% and 10.04% lower than SS-2 and SS-4 respectively. In MAXIMUM, the difference between the scaling strategies is negligible. (iii) Compared to the scaling strategies, FIXED that has 3000 aggregators throughout the experiment has on average 20.52% and 10.93% higher average relative error under UP-DOWN and DOWN-UP respectively. This is because of the higher number of aggregators (3000 over 1500) in the network.

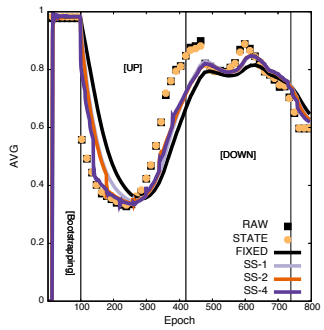
Figure 5 illustrates the computed aggregates over runtime. The plots include the following information: (i) RAW that is the 'true' values of the aggregates computed using as input the original data. (ii) STATE that is the 'true' values of the aggregates computed using as input the selected states extracted by clustering the original data. (iii) FIXED that is the estimated values of the aggregates computed by DIAS with the maximum of 3000 aggregators online. (iv) The estimated values of the aggregates using the three scaling strategies. STATE approximates very well the RAW in SUMMATION and AVERAGE, whereas in the high values of MAXIMUM the DIAS approximation deteriorates. The average relative error between RAW and STATE in MAXIMUM is 4.70%.

Moreover, Figure 5 explains the findings shown in Figure 4. For instance, at each scaling step during which aggregators are connected, there are sudden drops in accuracy, which are especially high in SUMMATION and MAXIMUM, given that AVERAGE has mainly values in $[0, 1]$. It is also observed that during the epochs 100 to 419 in which RAW (and therefore STATE) change rapidly to lower values (energy consumption decreases), the drops in accuracy are higher than the more

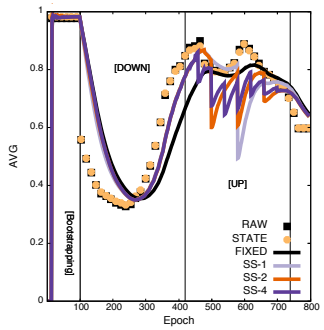


(a) SUMMATION, UP-DOWN

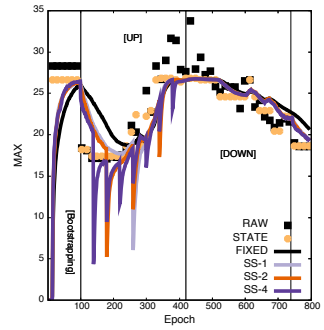
(b) SUMMATION, DOWN-UP



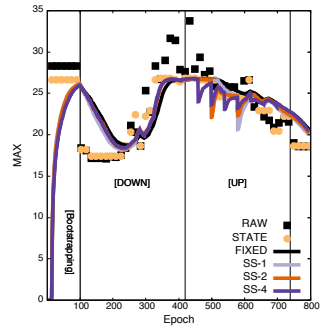
(c) AVERAGE, UP-DOWN



(d) AVERAGE, DOWN-UP

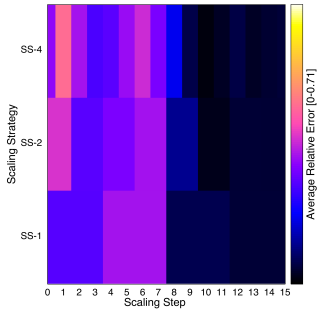


(e) MAXIMUM, UP-DOWN

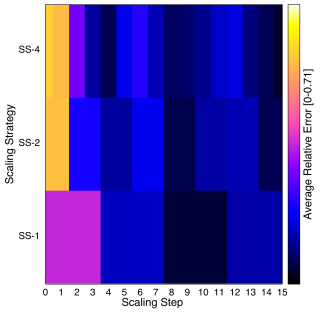


(f) MAXIMUM, DOWN-UP

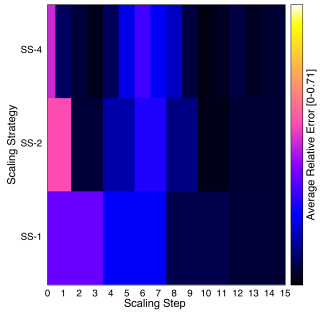
Figure 5. Computed aggregates over runtime.



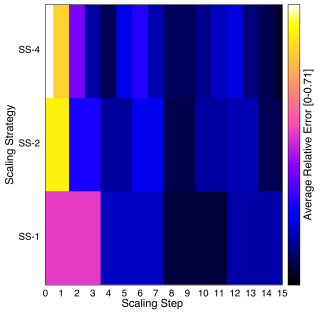
(a) SUMMATION, UP-DOWN



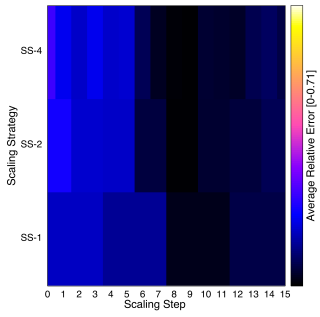
(b) SUMMATION, DOWN-UP



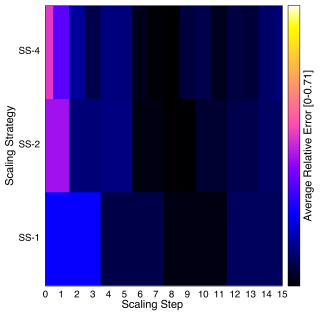
(c) AVERAGE, UP-DOWN



(d) AVERAGE, DOWN-UP



(e) MAXIMUM, UP-DOWN



(f) MAXIMUM, DOWN-UP

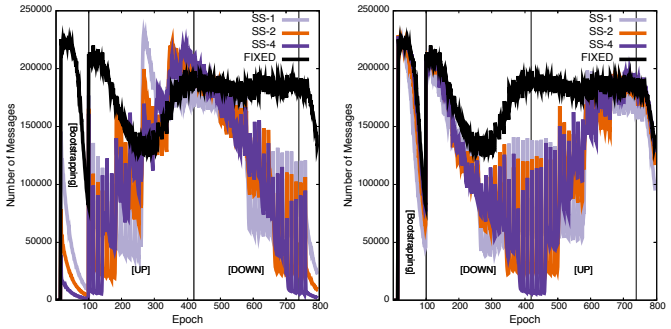
Figure 6. Accuracy of aggregation functions at each scaling step for different demand profiles, scaling strategies and demand profiles.

stable aggregates during the epochs 420 to 739. Drops in accuracy are higher when a higher number of aggregators is involved in the scaling step and therefore the drop peaks of SS-1 are higher than the ones of SS-2 and SS-4. These drops are quantified in Figure 6 that illustrates the average relative error for each scaling strategy and each scaling step. For instance, the average relative error of SUMMATION at scaling step 4 under UP-DOWN is 10.70% and 20.27% higher for SS-1 compared to SS-2 and SS-4. The respective error increase for AVERAGE is 32.74% and 63.49%.

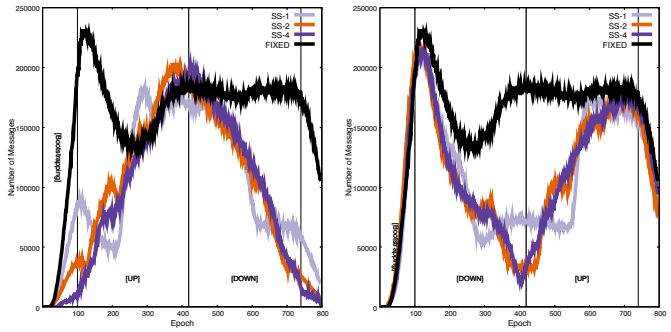
Note also that in both Figure 5 and 6 the scaling strategies have the capacity to react faster than FIXED in changes of the selected states (epochs 300 to 500 in Figure5a to 5d). This is because of the lower number of data consumers, resulting in a lower number of required aggregate updates in the network. However, the high accuracy in FIXED is more stable than the one of the scaling strategies in which the number of aggregators varies.

Figure 7 shows the communication cost of the scaling strategies in simulation and live simulation mode. The following observations can be made: (i) At each scaling step of the strategies in simulation mode, there is a burst in the communication cost as disseminators discover the joined aggregators and initiate aggregation sessions with them. This can be seen in Figure 7a and Figure 7b for UP-DOWN and DOWN-UP respectively. (ii) In live operational mode, this effect smooths out due to the asynchronicity of the scaling events. (iii) The communication cost of FIXED follows the pattern of the data, meaning the changes of SUMMATION as shown in Figure 5. The communication cost of FIXED in simulation and live mode is on average 32.54% and 34.25% higher than the one of the strategies as a higher number of aggregators is present in FIXED.

Figure 8 illustrates the total communication cost, aggregated



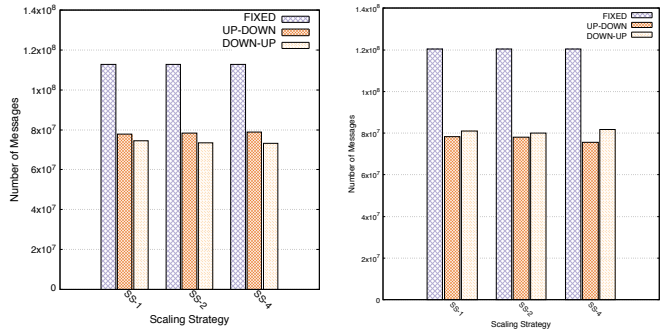
(a) UP-DOWN (b) DOWN-UP



(c) UP-DOWN (d) DOWN-UP

Figure 7. Communication cost in simulation and live operational mode for different scaling strategies and demand profiles.

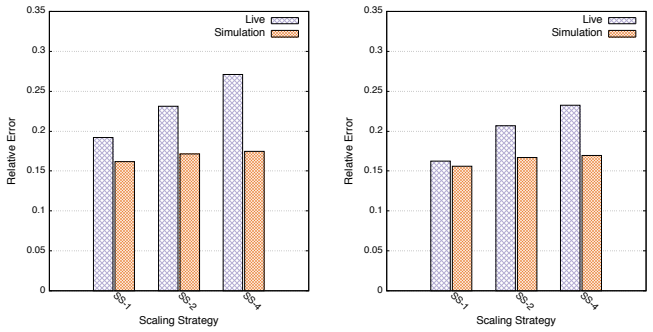
over all epochs, of the scaling strategies in simulation and live operational mode. The two plots indicate the following: (i) The three scaling strategies have on average the same communication cost as it is required for aggregators to exchange a fixed number of messages so that aggregates converge to the actual values, regardless of how aggregators join the network. (ii) The total communication cost of UP-DOWN and DOWN-UP between simulation and live mode is comparable. (iii) FIXED under the live mode has 6.4% higher communication cost than FIXED under simulation. (iv) The total communication cost under the live operational mode increases on average 4.4%, 3.8%, 3.0% in SS-1, SS-2, SS-4 respectively compared to simulation.



(a) Simulation (b) Live

Figure 8. Total communication cost in simulation and live operational mode for different scaling strategies and demand profiles.

Figure 9 illustrates the difference in accuracy between the simulation and live operational mode. On average, the relative error of SUMMATION in live is 34.21% and 37.25% higher than simulation under UP-DOWN and DOWN-UP respectively. This is because of the infrastructure lag and asynchronicity when aggregators join the network that alters the demand profiles as shown in Figure 3c and 3d.



(a) UP-DOWN (b) DOWN-UP

Figure 9. Accuracy of SUMMATION in simulation and live operational mode for the two demand profiles.

While in simulation under UP-DOWN the average relative error of SS-4 is 7.22% and 1.68% higher than SS-1 and SS-2 respectively, the respective increase for live is 29.13% and 14.57%. The higher the number of aggregators that simultaneously join the network, the higher the lag from asynchronicity effects that increases inaccuracies. This pattern is also observed in DOWN-UP. The average relative error of SS-4 in simulation is 7.88% and 1.25% higher than SS-1 and SS-2 respectively, while in live the respective increase is 30.0% and 10.89%.

V. Comparison with Related Work

To the best of authors' knowledge, there is no other related method on dynamic computational demand in decentralized in-network aggregation for a fair and meaningful quantitative comparison. Therefore, this section focuses on a qualitative comparison.

The majority of related work [22], [31], [32], [33] on decentralized in-network aggregation relies on static input values, a static network and static demand. Although a periodic reset of the aggregation process can encounter to a limited extent the aggregation dynamics, other research efforts [34], [21], [35], [14], [20], [23], [36] introduce aggregation mechanisms that are by-design resilient to node failures and changing of input values, which are both studied from the perspective of data suppliers. In contrast, this work studies rigorously decentralized in-network aggregation under dynamic computational demand, in which the participation of data consumers in the network varies.

Gossiping communication is introduced in an earlier model [37] of opportunistic crowd-sensing for smart cities. In this earlier model, cars, bicycles and pedestrians collect information in transit. Gossiping is used to increase the efficiency

of data transmission to stationary stations and therefore it is mainly applicable for dynamic data supply rather computational demand. Moreover, the system is restricted to sensing, in contrast to this work that additionally addresses the computation of aggregation functions. The Cloud of Things approach for sensing as a service [38] goes a step further by introducing an architecture for crowd-sensing and crowd-processing using gossiping communication for resource discovery. However, the data processing is designed for batch job execution rather than collective computations over a decentralized network.

In contrast to the reactive self-adaptive model studied in this paper, on-demand application-level video multicasting applies proactive optimal placement of content to improve system performance [39]. In the context of in-network aggregation, a more proactive approach may require accurate prediction models of the data consumers' participation as well as information about how the input data of the aggregation functions change. Encountering for both of these requirements is a challenge. The self-adaptation model introduced is totally data-independent and can react to any change in the computational demand.

Cloud computing environments often require dynamic resource allocation and provisioning to adapt to dynamic demand originated by the number of users, connections, and requests received [10], [11], [12]. Although resources and computations are distributed, management is usually centralized and therefore optimization techniques are employed for the resource allocation and provisioning.

VI. Conclusion and Future Work

This paper concludes that decentralized data analytics designed to be self-adaptive to dynamic computational demand can provide high accuracy, while communication cost is controlled and minimized. The nodes in a decentralized network running the proposed self-adaptation model become self-aware of the online data consumers in the network and therefore they can orchestrate the data sharing from data suppliers to data consumers in a peer-to-peer fashion. The model proves to be simple yet, agile and modular when applied in the decentralized in-network aggregation service of DIAS. No major architectural changes required for the applicability of the model. Experimental evaluation in simulation and a live deployment using data from a real-world smart grid pilot project confirm the cost-effectiveness of self-adaptation, in terms of accuracy and communication cost, under extreme variation in computational demand. Lessons learnt include the following: (i) The proposed self-adaptation process is more cost-effective when the network does not undergo major changes in the shared data. (ii) A lower number of perturbations in computational demand results in higher performance than a higher number of perturbations, even if fewer perturbations are more significant in scale. (iii) Live deployment approximates well the simulation results and the mismatches mainly originate from timing and synchronization issues of the deployment process. The contributions of this paper increase the technical readiness level of DIAS that aspires to turn fully decentralized and participatory data analytics into public good using crowd-sourced computational resources .

Future work concerns the applicability of the self-adaptation model to other systems beyond DIAS that require adaptation to varying computational demand such as cloud resource allocation. Real-world demand profiles of data analytics services will also provide an opportunity to evaluate the self-adaptation model in economic terms. Beyond aggregation functions, the findings of this paper can provide new insights on how to perform more complex analytics such as machine learning [40], [41] over large-scale and highly dynamic decentralized networks. The further deployment of DIAS to larger testbeds and network infrastructures in which users can participate with smart phones and Internet of Things platforms such as Nervousnet [42], [43], [2] is ongoing work.

Acknowledgment

This work is supported by the European Community's H2020 Program under the scheme 'INFRAIA-1-2014-2015: Research Infrastructures', grant agreement #654024 'SoBigData: Social Mining & Big Data Ecosystem' (http://www.sobigdata.eu), the European Community's H2020 Program under the scheme 'ICT-10-2015 RIA', grant agreement #688364 'ASSET: Instant Gratification for Collective Awareness and Sustainable Consumerism' (http://www.asset-consumerism.eu) and the European Research Council's Advanced Investigator Grant 'Momentum' with grant agreement #324247.

Special thanks go to Rok Roskar, Lorenz Blum and the ETH Scientific IT Services for their support to deploy DIAS on the Euler high performance cluster infrastructure of ETH Zurich.

References

[1] S. Hajian, F. Bonchi, and C. Castillo, "Algorithmic bias: From discrimination discovery to fairness-aware data mining," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 2125–2126.

[2] D. Helbing and E. Pournaras, "Society: Build digital democracy," *Nature*, vol. 527, pp. 33–34, 2015.

[3] S. Sevignani, "The problem of privacy in capitalism and the alternative social networking site diaspora," *tripleC: Communication, Capitalism & Critique. Open Access Journal for a Global Sustainable Information Society*, vol. 10, no. 2, pp. 600–617, 2012.

[4] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang, "Cdas: A crowdsourcing data analytics system," *Proc. VLDB Endow.*, vol. 5, no. 10, pp. 1040–1051, Jun. 2012.

[5] W. Willett, J. Heer, and M. Agrawala, "Strategies for crowdsourcing social data analysis," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 227–236.

[6] S. Shah, F. Bao, C.-T. Lu, and I.-R. Chen, "Crowdsafe: crowd sourcing of crime incidents and safe routing on mobile devices," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2011, pp. 521–524.

[7] Z. Xu, Y. Liu, N. Yen, L. Mei, X. Luo, X. Wei, and C. Hu, "Crowdsourcing based description of urban emergency events using social media big data," *IEEE Transactions on Cloud Computing*, 2016.

[8] E. Pournaras, M. Yao, and D. Helbing, "Self-regulating supply-demand systems," *Future Generation Computer Systems*, vol. 76, pp. 73 – 91, 2017.

[9] A. Bär, P. Casas, L. Golab, and A. Finamore, "Dbstream: an online aggregation, filtering and processing system for network traffic monitoring," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*. IEEE, 2014, pp. 611–616.

[10] M. Alrokayan, A. V. Dastjerdi, and R. Buyya, "Sla-aware provisioning and scheduling of cloud resources for big data analytics," in *Cloud Computing in Emerging Markets (CCEM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–8.

[11] F. Zulkernine, P. Martin, Y. Zou, M. Bauer, F. Gwadry-Sridhar, and A. Aboulnaga, "Towards cloud-based analytics-as-a-service (claaas) for big data analytics in the cloud," in *Big Data (BigData Congress), 2013 IEEE International Congress on*. IEEE, 2013, pp. 62–69.

[12] M. M. Hassan, B. Song, M. S. Hossain, and A. Alamri, "Qos-aware resource provisioning for big data processing in cloud computing environment," in *Computational Science and Computational Intelligence (CSCI), 2014 International Conference on*, vol. 2. IEEE, 2014, pp. 107–112.

[13] L. Nyers and M. Jelasity, "A comparative study of spanning tree and gossip protocols for aggregation," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 16, pp. 4091–4106, 2015.

[14] E. Pournaras, J. Nikolic, A. Omerzel, and D. Helbing, "Engineering democratization in internet of things data analytics," in *Proceedings of the 31st IEEE International Conference on Advanced Information Networking and Applications-AINA-2017*.

[15] J. He, D. J. Miller, and G. Kesidis, "Latent interest-group discovery and management by peer-to-peer online social networks," in *Social Computing (SocialCom), 2013 International Conference on*. IEEE, 2013, pp. 162–167.

[16] A. Osman, M. El-Refaey, and A. Elnaggar, "Towards real-time analytics in the cloud," in *Services (SERVICES), 203 IEEE Ninth World Congress on*. IEEE, 2013, pp. 428–435.

[17] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: A self-tuning system for big data analytics." in *CIDR*, vol. 11, 2011, pp. 261–272.

[18] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems (TOCS)*, vol. 25, no. 3, p. 8, 2007.

[19] E. Pournaras, "Multi-level reconfigurable self-organization in overlay services," Ph.D. dissertation, TU Delft, Delft University of Technology, 2013.

[20] E. Pournaras and J. Nicolic, "Self-corrective dynamic networks via decentralized reverse computations," in *Proceedings of the 14th International Conference on Autonomic Computing (ICAC 2017)*, July 2017.

[21] P. Jesus, C. Baquero, and P. S. Almeida, "Flow updating: Fault-tolerant aggregation for dynamic networks," *Journal of Parallel and Distributed Computing*, vol. 78, pp. 53–64, 2015.

[22] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems (TOCS)*, vol. 23, no. 3, pp. 219–252, 2005.

[23] F. Wuhib, M. Dam, R. Stadler, and A. Clem, "Robust monitoring of network-wide aggregates through gossiping," *IEEE Transactions on Network and Service Management*, vol. 6, no. 2, pp. 95–109, June 2009.

[24] E. Pournaras, J. Nikolic, P. Velásquez, M. Trovati, N. Bessis, and D. Helbing, "Self-regulatory information sharing in participatory social sensing," *EPJ Data Science*, vol. 5, no. 1, p. 1, 2016.

[25] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[26] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, "Protopeer: a p2p toolkit bridging the gap between simulation and live deployement," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 60.

[27] R. Kota, G. Chalkiadakis, V. Robu, A. Rogers, and N. R. Jennings, "Cooperatives for demand side management," in *Proceedings of the 20th European Conference on Artificial Intelligence*, ser. ECAI'12. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2012, pp. 969–974.

[28] M. Vasirani and S. Ossowski, "A collaborative model for participatory load management in the smart grid," in *Workshop on AI Problems and Approaches for Intelligent Environments*, 2012, p. 21.

[29] T. A. Rodden, J. E. Fischer, N. Pantidi, K. Bachour, and S. Moran, "At home with agents: exploring attitudes towards future smart energy infrastructures," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 1173–1182.

[30] E. Pournaras, M. Vasirani, R. E. Kooij, and K. Aberer, "Measuring and controlling unfairness in decentralized planning of energy demand," in *Energy Conference (ENERGYCON), 2014 IEEE International*. IEEE, 2014, pp. 1255–1262.

[31] S. Kashyap, S. Deb, K. Naidu, R. Rastogi, and A. Srinivasan, "Efficient gossip-based aggregate computation," in *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2006, pp. 308–317.

[32] M. Haridasan and R. Van Renesse, "Gossip-based distribution estimation in peer-to-peer networks." in *IPTPS*. Citeseer, 2008, p. 13.

[33] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 2, p. 7, 2008.

[34] O. Kennedy, C. Koch, and A. Demers, "Dynamic approaches to in-network aggregation," in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE, 2009, pp. 1331–1334.

[35] N. Ahmed, D. Hadaller, and S. Keshav, "Incremental maintenance of global aggregates," UW. Technical Report CS-2006-19, University of Waterloo, ON, Canada, Tech. Rep., 2006.

[36] D. Pianini, J. Beal, and M. Viroli, "Improving gossip dynamics through overlapping replicates," in *International Conference on Coordination Languages and Models*. Springer, 2016, pp. 192–207.

[37] M. C. Guenther and J. T. Bradley, "On performance of gossip communication in a crowd-sensing scenario," in *International Conference on Quantitative Evaluation of Systems*. Springer, 2014, pp. 122–137.

[38] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Cloud of things for sensing as a service: sensing resource discovery and virtualization," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–7.

[39] B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 2, pp. 566–579, 2013.

[40] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 556–571, 2013.

[41] A. Berta, I. Hegedus, and M. Jelasity, "Dimension reduction methods for collaborative mobile gossip learning," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, Feb 2016, pp. 393–397.

[42] E. Pournaras, I. Moise, and D. Helbing, "Privacy-preserving ubiquitous social mining via modular and compositional virtual sensors," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. IEEE, 2015, pp. 332–338.

[43] F. Musciotto, S. Delpriori, P. Castagno, and E. Pournaras, "Mining social interactions in privacy-preserving temporal networks," in *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*. IEEE, 2016, pp. 1103–1110.