
Proceedings Paper:
Pournaras, E and Nikolic, J (2017) Self-Corrective Dynamic Networks via Decentralized Reverse Computations. In: 2017 IEEE International Conference on Autonomic Computing (ICAC). 2017 IEEE International Conference on Autonomic Computing (ICAC), 17-21 Jul 2017, Columbus, OH, USA. IEEE, pp. 11-20. ISBN: 978-1-5386-1762-5. EISSN: 2474-0756.

https://doi.org/10.1109/icac.2017.30

Self-corrective Dynamic Networks via Decentralized Reverse Computations

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

***Abstract*—The feasibility of large-scale decentralized networks for local computations, as an alternative to big data systems that are often privacy-intrusive, expensive and serve exclusively corporate interests, is usually questioned by network dynamics such as node leaves, failures and rejoins in the network. This is especially the case when decentralized computations performed in a network, such as the estimation of aggregation functions, e.g. summation, are linked to the actual nodes connected in the network, for instance, counting the sum using input values from only connected nodes. Reverse computations are required to maintain a high aggregation accuracy when nodes leave or fail. This paper introduces an autonomic agent-based model for highly dynamic self-corrective networks using decentralized reverse computations. The model is generic and equips the nodes with the capability to disseminate connectivity status updates in the network. Highly resilient agents to the dynamic network migrate to remote nodes and orchestrate reverse computations for each node leave or failure. In contrast to related work, no other computational resources or redundancy are introduced. The self-corrective model is experimentally evaluated using real-world data from a smart grid pilot project under highly dynamic network adjustments that correspond to catastrophic events with up to 50% of the nodes leaving the network. The model is highly agile and modular and is applied to the large-scale decentralized aggregation network of DIAS, the Dynamic Intelligent Aggregation Service, without major structural changes in its design and operations. Results confirm the outstanding improvement in the aggregation accuracy when self-corrective actions are employed with a minimal increase in communication overhead.**

***Keywords*-self-correction; adaptation; accuracy; reverse computation; data analytics; decentralized network; aggregation; agent; migration; robustness; fault-tolerance**

I. Introduction

The pervasiveness and increasing computational capacity of smart Internet of Things devices equipped with networking capabilities allow complex distributed computations to be performed over networks, for instance, sensor networks computing the spread of oil spills [1], smart grids measuring power peaks in energy demand [2] or monitoring of automotive traffic [3]. Decentralized computations over dynamic networks are highly challenging to perform accurately and fast under changing input data, nodes temporarily leaving, failing or rejoining the network [4], [5], [6], [7], [8]. However, algorithms for computations in decentralized networks are by design more privacy-preserving, scalable, respect users' autonomy and do not require significant investments in expensive big data computational resources [9], [10], [11], [12].

This paper studies a complex and challenging problem of decentralized computations: when computations performed in each node of a network are linked to the connectivity status of all other nodes, adaptive or corrective computations are required to roll-back computational results to the latest network status in terms of connected nodes. However, orchestrating adaptation and corrective actions is even more challenging when nodes leave the network as there are lower computational resources in the network to employ for this purpose. For example, consider the problem of decentralized in-network aggregation [7], [13]: each node in a network computes an aggregation function, e.g. summation, given an input value from every connected node. If a node leaves the network, all summations computed with input from this leaving node need to be reverted by subtracting its value. Such rollback actions are referred to in this paper as *reverse computations* and they are known for their significance in efficient distributed systems [14], [15], [16], [17]. The goal of this work is to design autonomic dynamic networks that are self-corrective using decentralized reverse computations.

This paper introduces a new agent-based self-corrective model for dynamic computational networks, which, in contrast to related work [18], [19], [8], [12], does not rely on replication servers, proxies, storage of checkpoints or big data analytics for fault analysis. Each node in the network creates two software agents, the *status* and the *corrective* agent. The status agent publishes connectivity status information about the parent node. The corrective agent migrates to other host nodes from which it monitors the connectivity status of its parent. If the parent node leaves or fails, its remote corrective agent initiates and orchestrates reverse computations with the other nodes in the network. If the parent node reconnects, the corrective agent terminates its operations and has the option to migrate back to the parent agent to update the parent's outdated state for the time period it has been disconnected. A fully decentralized gossip-based communication supports the information exchange and dissemination between the status and the corrective agent.

The applicability of the proposed self-corrective model is studied in decentralized in-network aggregation using the DIAS system [20], [7]. The capability of DIAS to reverse computations when nodes change the input values in the ag-

gregation functions is extended to nodes leaving and rejoining the network. In contrast to related work in which agility comes as a trade-off to resilience [21], the introduction of the generic and modular self-corrective model in DIAS does not require any major structural changes in its design. Extensive experimental results verify the improvement potential of the aggregation accuracy by the proposed self-corrective model using real-world data from a smart grid pilot project. Evaluation under lightweight and heavyweight network adjustments corresponding to catastrophic events with up to 50% of the nodes leaving the network provides the proof-of-concept. Results show significant improvement in accuracy, while the corrective agents manage to tolerate the network dynamics via consecutive migrations. They eventually perform the required reverse computations in a symbiotic and autonomic fashion, while communication cost remains low.

The main contributions of this paper are outlined as follows:

- The introduction of a new agent-based self-corrective model for orchestrating decentralized reverse computations in large-scale dynamic networks.
- The expansion of the DIAS functionality with reverse computations in dynamic networks in which nodes temporarily leaving and rejoining.
- Verification of the improvement potential that reverse computations have on the aggregation accuracy under lightweight and heavyweight network adjustments.

This paper is outlined as follows: Section II formulates the research problem. Section III introduces the self-corrective model for reverse computations in dynamic networks. Section IV illustrates the applicability of the self-corrective model in the decentralized in-network aggregation of DIAS. Section V experimentally evaluates the self-corrective model in lightweight and heavyweight scenarios of nodes temporarily leaving and rejoining the network. Section VI compares the self-corrective model with related work. Finally, Section VII concludes this paper and outlines future work.

II. Decentralized Reverse Computations

Assume the extreme scenario of a decentralized dynamic network with n nodes each with a local state $s_i \in \mathbb{R}$. The network is dynamic as nodes may *temporary leave*, *fail* or *rejoin* during the overall system runtime. Each node $i \in \{1, ..., n\}$ of the network performs a number of n incremental operations $a_{i,j} = f_{\circlearrowleft}(s_j, a_{i,j-1})$ using the input state of every node $j \in \{1, ..., n\}$ and assuming that $a_{i,0} = 0$, for instance. Given that the network is decentralized, nodes need to discover each other and establish a remote peer-to-peer interaction to exchange their local state s_i and s_j. Gossiping protocols provide effective node discovery in dynamic decentralized networks [22]. The sequence $(a_{i,j})_{j=1}^{n}$ of n updates completes the required operations for each node i. Therefore, the presence of a node in the network determines the required computations of the other nodes present in the network.

However, assume without loss of generality that the jth node leaves the network or fails, reverse computations are required to roll back the result of the earlier computation $a_{i,j} = f_{\circlearrowleft}(s_j, a_{i,j-1})$. Such a reverse computation is defined as $a_{i,j-1} = f_{\circlearrowleft}(s_j, a_{i,j})$, assuming that the two operations $f_{\circlearrowleft}()$ and $f_{\circlearrowleft}$ are *constructive* in nature as defined by the fact that they require no history and only the most current values of the variables can undo the primary operation [14]. Examples of such operations are the $++$, $--$, $+=$, $-=$, $*=$ and $/=$, with the two latter ones requiring special treatment in case of multiplication or devide by zero as well as overflow and underflow conditions. Reversible operations are significant and common for several computations in distributed systems and decentralized networks, for instance, in-network aggregation for maintaining accuracy [20], parallel simulations for efficiency in memory utilization [14], fault-tolerance in large parallel systems [15] as well as data analytics and big data scientific applications [16], [17].

Given this setting, a *self-corrective* system property for a decentralized network with a varied number of participating nodes due to leaves, failures or rejoins is defined as follows:

Definition 1. *A dynamic decentralized network is defined as* self-corrective *if for each node $i \in \{1, .., n\}$ performing once and only once all sequence operations $(a_{i,j})_{j=1}^{n}$, where $a_{i,j} = f_{\circlearrowleft}(s_j, a_{i,j-1})$, $\forall j \in \{1, .., n\}$, a respective reverse computation $a_{i,j-1} = f_{\circlearrowleft}(s_j, a_{i,j})$, $\forall i \in \{1, .., n\}$ and $\forall j \neq i \ni \{1, .., n\}$ is performed if and only if the jth node does not anymore participate in the network, while node i participates.*

Without loss of generality, reverse computations are applicable for any node j. This paper focuses on the design, implementation and evaluation of an autonomic mechanism that builds dynamic networks that are self-corrective by design according to Definition 1.

III. Self-corrective Dynamic Networks

This paper introduces an agent-based model for self-corrective computations in dynamic decentralized networks. The model is designed with two agents that every node constructs to collectively form a self-corrective network: (i) the *status agent* and (ii) the *corrective agent*. Figure 1 illustrates a representative lifecycle of the proposed agent-based model.

The status agent is responsible for reporting the connectivity status of the node: (i) *rejoin*, (ii) *leave* and (iii) *failure*. The statuses of the node are disseminated in the network via the peer sampling service by using the *node descriptor*. The node descriptor contains base information such as the IP address, the port number, the descriptor age, as well as registered application information that can be carried within the descriptor. All necessary information is periodically disseminated via gossip exchanges of the node descriptor according to the execution period of the peer sampling service. The failure status is actually deductive: the status agent does not report an actual failure but rather that is not failed, given that a node failure terminates the status agent as well. The failure status is derived from the age of the descriptor. At every gossip exchange, a node i resets the age of its descriptor back to zero. The follow up gossip exchanges by the other nodes increase the

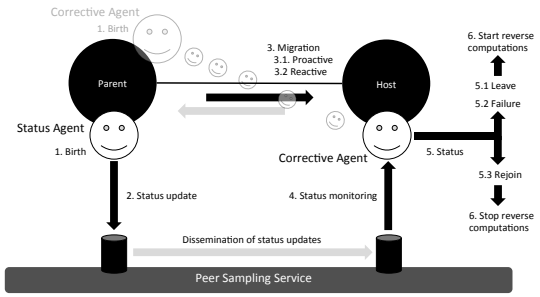


Figure 1. The proposed self-corrective agent-based model for dynamic networks. Each parent node creates a status and corrective agent. The latter migrates proactively or reactively in another host node. The status agent registers status information in the network via the peer sampling service. The migrated corrective agent can remotely receive this status information and trigger reverse computations when the parent node leaves or fails. If the node rejoins the network, the reverse computations are terminated.

age descriptor. The disseminated descriptors of a node i in the network have on average a low age value if node i is connected and can reset the age descriptor. In contrast, a disconnected node does not perform gossip exchanges and therefore does not update the age of the disseminated descriptors. The earlier work [22] on the peer sampling service provides further information about the healing process of gossiping using the age descriptors. All in all, the status agent remains local and it has minimal tasks for execution.

The corrective agent is responsible for orchestrating the reverse computations. It operates remotely by migrating in a secure way [23] from the *parent node* that creates it to another connected node, the *host node*. Migration is performed (i) *proactively*, so that the corrective agent does not terminate when the parent node fails or (ii) *reactively*, as a result of the parent node leaving on-demand. These are the two *migration modes* of the corrective agent illustrated in Algorithm 1.

Algorithm 1 Proactive and reactive migration of the corrective agent.

```
Require: migration mode, peer sampling service
loop
  if migration mode is 'proactive' then
    break
  else
    // migration mode is 'reactive'
    if local node leaves then
      break
    end if
  end if
end loop
get random node j from peer sampling service
migrate to host j
```

Consecutive migrations can be performed in case the host node is disconnected as well. In this case, a second proactive migration may represent the relocation of the corrective agent to another more reliable host agent, for instance, a host agent with a lower age descriptor value. A second reactive migration may represent the leaving of the host agent from the network.

The main operation of the corrective agent after it migrates to the host node is to monitor the network for status updates from its parent node. The corrective agent checks whether the node descriptor of the parent node is present in the *view* of the peer sampling service after every view update. The view is a buffer list of limited size containing the node descriptors received from other nodes via gossiping. If the parent node descriptor is present with a status 'leave' or 'failure', then reverse computations are initiated. If the status is 'rejoin', reverse computations that are in-progress are terminated. The failure status is not explicit and can be detected by an overpass of a threshold[1] in the age descriptor of the parent node. Algorithm 2 illustrates the main operations of the corrective agent in the host node.

Algorithm 2 Main operations of the corrective agent in the host node.

```
Require: parent i, migration type, peer sampling service
for every view update in the peer sampling service do
  if view has parent descriptor then
    if parent descriptor status is 'failure' or 'leave' then
      start reverse computations
    else
      if parent descriptor status is 'rejoin' then
        stop reverse computations
        if migration type is stateful then
          migrate to parent i
        end if
      else
        // parent descriptor status is 'connected'
      end if
    end if
  end if
end for
```

Depending on the computations $f_{\circlearrowleft}()$, $f_{\circlearrowleft}()$ of an application and the network scenario, the corrective agents may transfer some information from the parent node to the host node and back to the parent node, after reverse computations are terminated, so that they guarantee a self-corrective network according to Definition 1. This defines a *stateful* migration, in contrast to a *stateless* migration in which the corrective agents do not need to return back to the parent agent rejoining in order the latter one to continue its operations. If an application requires explicit historical information about each node j involved in a computation $a_{i,j} = f_{\circlearrowleft}(s_j, a_{i,j-1})$, for instance, to prevent a double-counting in aggregation [20], [5], [7], a stateful migration is required.

The proposed multi-agent system equips dynamic decentralized networks with self-healing autonomic properties as the self-corrective reverse computations are symbiotically performed by the inner nodes of the network without introducing

[1]The threshold can be selected empirically by an analysis of the uptime and downtime distributions of the nodes in the network [24], [25].

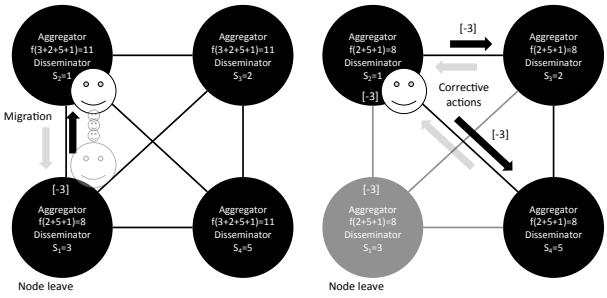
additional computational resources and redundancy, that are common in related work [26], [27]. The virtue of this design approach comes along with some challenges. The overall system design does not guarantee a self-corrective network given all uncertainties of large-scale dynamic decentralized networks. Catastrophic failures can result in the loss of corrective agents. Moreover, highly frequent join and leaves in the network may turn out reverse computation to be infeasible if the required convergence time of the peer sampling service is slower than the observed network dynamics [22]. Moreover, note that the peer sampling service is a highly robust decentralized protocol, yet it is a probabilistic mechanism. The goal of this paper is to quantify system performance and draw conclusions[2] about how large-scale decentralized dynamic networks can maximize the cost-effectiveness of their self-corrective operations.

IV. Decentralized Aggregation with Reverse Computations

As a proof of concept, the proposed self-corrective model is realized in a challenging application scenario of decentralized in-network aggregation. The problem statement remains as illustrated in Section II. The operations $f_{\circlearrowleft}()$ and $f_{\circlearrowleft}$ computed are aggregation functions, such as SUMMATION, AVERAGE, MAXIMUM, MINIMUM, etc. Reversed computations in aggregation functions[3] are crucial for maintaining a high accuracy when the local states of the nodes change values. By first performing reverse computations with the old values and then aggregating the new values, accuracy is maintained. DIAS, the *Dynamic Intelligent Aggregation Service* [20], [7] is designed to perform accurate in-network aggregation even when the local state $s_i = p_{i,s} \in \{p_{i,1}, ..., p_{i,k}\}$ of a node i changes among a finite set of k *possible states*. This is achieved via reverse computations orchestrated by a distributed efficient memory system of probabilistic data structures, the *bloom filters* [28]. Although this powerful capability of DIAS is exceptional among very few other related methodologies on decentralized in-network aggregation [13], [5], [6], it cannot be applied when nodes leave and rejoin the network as well. The proposed self-corrective model fills this gap by extending the capability of DIAS to perform reverse computations in dynamic networks with nodes leaving and rejoining.

The self-corrective model is the new contributed feature of DIAS and it is applied as follows. Each node i in DIAS may have an *aggregator*, a *disseminator* or both. The disseminator is a software agent that has information about the possible states $(p_{i,u})_{u=1}^{k}$, the local selected state $p_{i,s} = s_i$ and historic information forming the distributed memory system. The aggregator is also a software agent that computes the aggregation functions and performs reverse computations. It stores the output of the aggregation functions and shares part of the distributed memory system with the disseminators. Aggregation is performed as follows: disseminators discover aggregators using the peer sampling service and share their latest local selected state $p_{i,s} = s_i$. Therefore, the introduced self-corrective model reuses the peer sampling service of DIAS that is already part of its design. A peer-to-peer remote interaction between a disseminator and an aggregator that results in a potentially new incremental computation of an aggregation function is defined as an *aggregation session*.

Given that the disseminator is the agent that initiates the aggregation sessions in the DIAS network, the disseminator extends all functionality of the corrective agent. It is then capable of stateful migrations to host nodes when its parent nodes leaves the network. A migrated disseminator[4] to a remote host initiates aggregations sessions that involve reverse computations as shown in Figure 2. It is these aggregation sessions performed between migrated disseminators and aggregators that create a highly self-corrective DIAS network according to Definition 1. Migrations are stateful as DIAS relies on the distributed memory system of bloom filters[5]. Therefore, when a parent node rejoins, the disseminator migrates back with consistent information about all reverse computations performed during the time period the parent node has been disconnected.



(a) Migration. (b) Corrective actions.

Figure 2. An example of how aggregation accuracy is preserved in DIAS via migration and corrective actions. When a disseminator with a selected state of value 3 leaves the network, the corrective agent migrates to another host to apply corrective actions. In this example, the value 3 is removed from the sum of all other nodes in the network.

The reverse computations are implemented within the main DIAS runtime of Algorithm 1, 2 and 3 illustrated in earlier work [7]. The new features are the following: (i) Algorithm 1 is executed by the migrated disseminator that selects aggregators with which an aggregation session has been earlier performed (classified as exploited or outdated but not unexploited, line 5). (ii) The computation of the aggregation functions in Algorithm 2 is an actual reverse computation as defined in Section II. (iii) The disseminator and aggregator involved in

[2]Such measurements and conclusions can be used to design empirical heuristics, e.g. decision trees, to customize system performance as earlier shown [2], [7]. The design of such heuristics is out of the scope of this paper.

[3]Although MAXIMUM and MINIMUM are not constructive in nature, reverse computations can be applied using the TOP-K and BOTTOM-K respectively as shown in earlier work [7]. In this case, a memory buffer of size k supports the reverse computations under changes in local states.

[4]It is implemented in DIAS as a serializable object transfered via an Apache MINA TCP socket.

[5]Bloom filters offer both space savings and privacy-preservation as no explicit historic information is migrated [7]. The self-corrective model does not result in additional information reveal.

the aggregation session are removed from the memory by decreasing the counters of the counting bloom filters (line 2 and 1 of Algorithm 1 and 2 respectively).

The introduction of the self-corrective model in DIAS requires almost no changes in its design and main operations. The distributed memory system remains intact: no new bloom filters are introduced, consistency in the computations is guaranteed via the existing bloom filters used in the very initial design of DIAS [20]. The same exactly holds for the interactions: an aggregation session is performed in the same way whether a corrective agent is employed or not. The additional messages exchanged for the migration of disseminators is functionality inherited from the introduced corrective agent.

V. Experimental Evaluation

The self-corrective model in DIAS is implemented in Java using the Protopeer distributed prototyping toolkit [29]. A Protopeer implementation[6] of the peer sampling service is part of the DIAS design and it is reused by the self-corrective model. Both DIAS and the peer sampling service are deployed in the Euler[7] HPC cluster infrastructure of ETH Zurich.

DIAS is fed with real-world data[8] from ECBT, the *Electricity Customer Behavior Trial* that is a state-of-the-art pilot project about the electricity consumption in Ireland. The project ran during the period 2009-2010 with 6435 residential and small-medium enterprise consumers, from which 3000 residential consumers are used for shorter execution times of the experiments. Consumption data are collected from smart meters every 30 minutes. The data of 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[9].

The epoch duration is selected as 30 minutes/14 DIAS executions=2.14 minutes (2.14/4=0.5 minutes for the peer sampling service) to match the data records used. A high execution rate of DIAS improves convergence speed but also increases the communication rate, which is though minimized to zero after convergence. Each experiment runs for 800 epochs in total. 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 swap parameter of 24 and a healer parameter of 1 [13]. Each of the 3000 nodes of DIAS is equipped with both a disseminator and an aggregator to test the most demanding scenario. A maximum of 40 aggregation sessions per epoch are initiated by each disseminator. Results for the AVERAGE, SUMMATION and MAXIMUM aggregation functions are presented.

The goal of the experimental evaluation is to show how the self-corrective model performs in DIAS under a *lightweight and heavyweight* scenario of network adjustments focusing[10] on *leaves and rejoins* of nodes. In both scenarios, the specific nodes leaving the network are fixed between the different experiments to compare the results. In the lightweight scenario, the number of connected nodes in the network are incrementally varied by a maximum of 20% as follows:

1) *UP-DOWN*: During bootstrapping, 20% of the nodes leave the network. In the first half of the aggregation time (the first 336 epoch after bootstrapping), nodes incrementally rejoin the network. The reverse process follows with 20% of the nodes incrementally leaving the network for the second half of the aggregation time.
2) *DOWN-UP*: After bootstrapping and during the first half of the aggregation time, meaning the first 336 epoch after bootstrapping, 20% of the nodes incrementally leave the network. The reverse process follows for the rest of the second half of the aggregation time with all left nodes incrementally rejoining the network.

The incremental leaves and rejoins are uniformly distributed in each UP and DOWN phase, yet, they are performed in varying batches of leaving and rejoining nodes, the *departure steps* (DS). The following departure steps are defined: (i) DS-75, (ii) DS-120, (iii) DS-200 and (iv) DS-300. DS-300 completes the leaves and rejoins of 20% (600) of the nodes in 2 steps, whereas, DS-75 performs 8 steps. Figure 4a and 4b illustrate the number of connected nodes in the network for UP-DOWN, DOWN-UP and varying departure steps.

In contrast, the heavyweight scenario of network adjustments incrementally removes 50% (1500) of the nodes in the network with varying speed referred to as *departure period*. The following departure periods are used in the experiments: (i) DP-10, (ii) DP-15 and (iii) DP-20. The number of nodes removed in each incremental step is controlled by the departure steps, as in the case of the lightweight scenario. In the heavyweight scenario, the following departure steps are used: (i) DS-50, (ii) DS-100, (iii) DS-150 and (iv) DS-250. Figure 8a to 8b show the number of connected nodes in the network for varying departure steps and periods.

The following measurements characterize the performance of the self-corrective model on DIAS:

- *Accuracy*: This is the average relative error of the aggregation functions in DIAS, defined by the absolute difference between actual and estimated values divided by the actual values. The accuracy evaluates the effectiveness of the self-corrective network according to Definition 1.
- *Communication cost*: The total number of DIAS messages and the number of messages originated by the migrations are measured. The communication cost of the peer sampling service is not shown as it is constant and it is governed by the execution period.
- *Rate of migration success*: This is the number of successful (consecutive) migrations divided by the total number

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

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

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

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

[10]Due to space limitations the focus is on nodes leaves and rejoins. Failure scenarios simply require activation of proactive migrations. Therefore all findings shown in this paper indicate the model performance under failures.

of migrations initiated in each epoch. An unsuccessful migration occurs when the local node of the corrective agent has to leave the network, the agent chooses a new random host node from the peer sampling service to migrate, however, this new host node has already left the network and as a result, the corrective agent is lost.

The rest of this section illustrates the experimental results under the lightweight and heavyweight network adjustments.

A. The improvement potential of accuracy

Table I and II verify the improvement potential in the accuracy of DIAS by the self-corrective model. A self-corrective DIAS network is compared to a network in which no corrective actions are employed, yet all input values from the connected nodes are counted in the aggregation functions. These are the actual 'true' values. Errors originated from non-convergence are excluded and the evaluation focuses on errors coming from node leaves. Results are shown for the SUMMATION that is highly influenced by node leaves. Results for the other aggregation functions confirm the conclusions of this section. The measurements of the relative errors are counted *after the computation of the aggregation functions is converged and at the epochs when nodes leave the network*. These are the epochs 500-800 and 250-400 in the UP-DOWN and DOWN-UP phases of the lightweight scenario respectively (see Figure 4a and 4b). In the heavyweight scenario, errors are measured after the 250th epoch and as long as the network size remains lower than 3000 (see Figure 8a and 8b).

Table I
AVERAGE RELATIVE ERRORS UNDER LIGHTWEIGHT NETWORK ADJUSTMENTS IN A NON-CORRECTIVE VS. A SELF-CORRECTIVE NETWORK OF DIAS.

Departure Step	Adjustment	Non-corrective	Self-corrective
DS-75	UP-DOWN	0.17	0.07
DS-75	DOWN-UP	0.19	0.18
DS-120	UP-DOWN	0.17	0.07
DS-120	DOWN-UP	0.19	0.18
DS-200	UP-DOWN	0.16	0.07
DS-200	DOWN-UP	0.20	0.18
DS-300	UP-DOWN	0.15	0.09
DS-300	DOWN-UP	0.20	0.16

Table II
AVERAGE RELATIVE ERRORS UNDER HEAVYWEIGHT NETWORK ADJUSTMENTS IN A NON-CORRECTIVE VS. A SELF-CORRECTIVE NETWORK OF DIAS.

Departure step	Departure period	Non-corrective	Self-corrective
DS-50	DP-10	0.16	0.12
DS-50	DP-15	0.28	0.13
DS-50	DP-20	0.43	0.10
DS-100	DP-10	0.25	0.12
DS-100	DP-15	0.22	0.12
DS-100	DP-20	0.17	0.12
DS-150	DP-10	0.25	0.11
DS-150	DP-15	0.25	0.12
DS-150	DP-20	0.24	0.13
DS-250	DP-10	0.25	0.09
DS-250	DP-15	0.25	0.10
DS-250	DP-20	0.25	0.10

Results confirm the significant improvement potential of accuracy. A self-corrective network decreases the errors 29% and 55% on average in the lightweight and heavyweight scenarios respectively.

B. Lightweight network adjustments

Figure 3 illustrates the average accuracy of the self-corrective model on DIAS under lightweight network adjustments. The following key observations are made: (i) In SUMMATION, the average relative error decreases by 6.46% and 15.23% during the UP phases by increasing the departure step from DS-75 to DS-300, while it increases by 20.27% and 3.19% during the DOWN phases. (ii) AVERAGE shows the same, but less significant trend with the respective values of 3.36% and 7.99% during UP and 5.57% and 1.28% during DOWN. (ii) In MAXIMUM, the average relative error decreases in all individual phases. The accuracy of SUMMATION over runtime is illustrated in Figure 4c and 4d. The baseline performance of FIXED refers to DIAS operating with all the 3000 nodes connected.

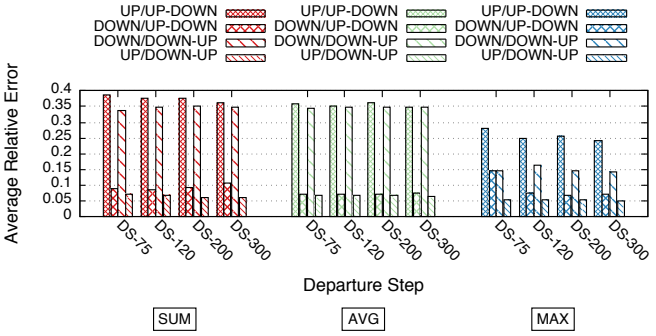
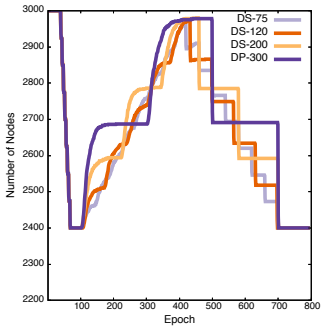


Figure 3. Average accuracy of the aggregation functions under leightweight network adjustments.

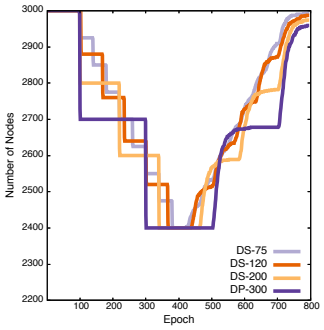
Figure 4e and 4f illustrate the number of messages exchanged for varying departure steps. The number of messages for UP-DOWN and DOWN-UP decreases 9.28% and 9.09% on average compared to FIXED despite the additional messages consumed for the migrations when nodes leave and rejoin. This is also confirmed in Figure 5a. The decrease in the communication cost is maximized when the number of nodes reaches the minimum during the DOWN phases. There are certain time points in which the communication costs under leaves and rejoins overpasses FIXED. This is when the network recovers its full size, while migrations that are still in progress cause the additional communication overhead.

Figure 5a illustrates the total number of messages aggregated over runtime. DS-300 has on average 2.89% lower communication cost than DS-120 in DOWN-UP, whereas, it is 1.41% higher in the UP-DOWN phase. The UP phase of UP-DOWN has the communication cost level of the DOWN phase in DOWN-UP. Respectively, the same holds for the DOWN phase of UP-DOWN and the UP phase of DOWN-UP.

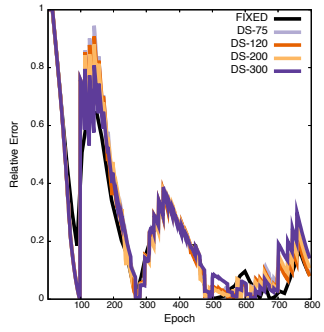
Figure 6a and 6b illustrate the communication cost originated from the consecutive migrations of the corrective agents.



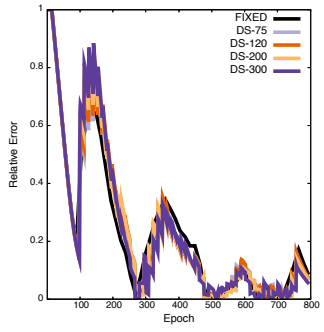
(a) Connected nodes, UP-DOWN.



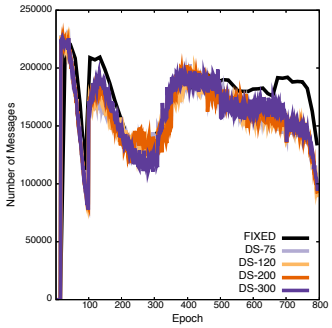
(b) Connected nodes, DOWN-UP.



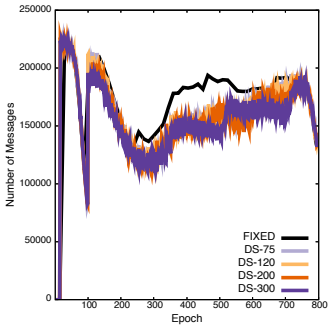
(c) SUMMATION accuracy, UP-DOWN.



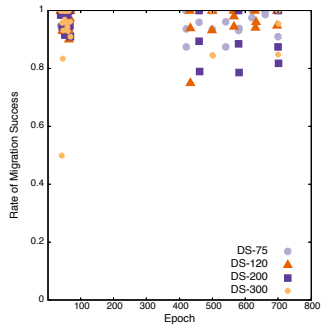
(d) SUMMATION accuracy, DOWN-UP.



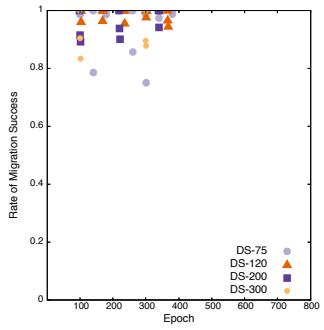
(e) Total number of messages, UP-DOWN.



(f) Total number of messages, DOWN-UP.



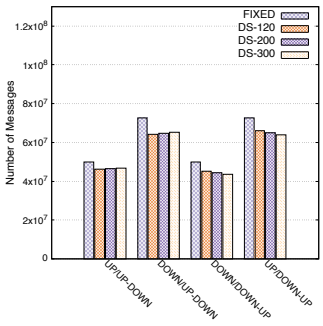
(g) Rate of migration success, UP-DOWN.



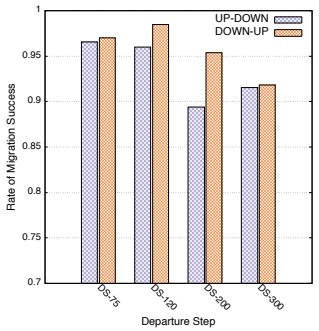
(h) Rate of migration success, DOWN-UP.

Figure 4. Performance of the self-corrective model on DIAS over its runtime under leightweight network adjustments. (a)-(b) Number of nodes in the network. (c)-(d) Accuracy of SUMMATION. (e)-(f) Total number of messages. (g)-(h) Rate of migration success.

Migrations back to the parent nodes are indicated in the plots



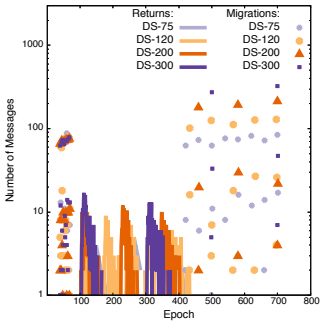
(a) Total number of messages.



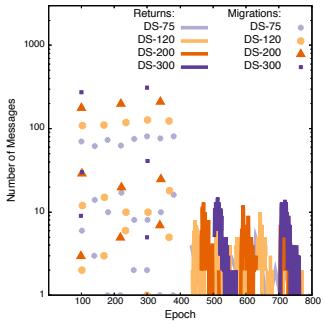
(b) Rate of migration success.

Figure 5. Total number of messages and rate of migration success aggregated over runtime under leightweight network adjustments.

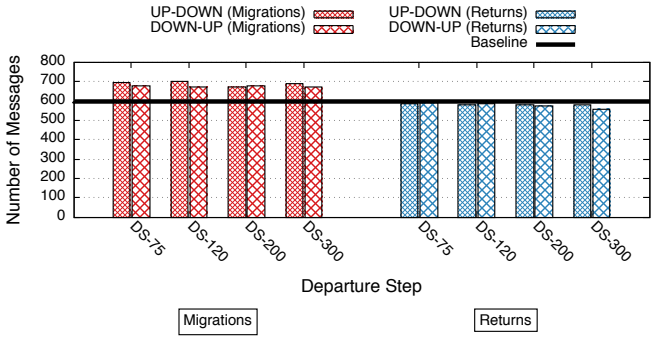
as 'returns'. The peaks in the plots correspond to the steps of node rejoins and the return migrations back to the parent nodes. Although the number of migrations from parent to host and from host to parent should be approximately the same[11] as shown in Figure 6c, the plots show that returns are highly spread over time given the convergence delay for discovering that rejoining parent nodes. In contrast, the migrations from parent to host happen in steps that justify the fewer samples with higher values appearing in the plots.



(a) Number of messages from migrations, UP-DOWN.



(b) Number of messages from migrations, DOWN-UP.



(c) Number of messages originated from migrations aggregated over runtime.

Figure 6. The communication cost of migrations under leightweight network adjustments.

Figure 6c illustrates the number of messages originated from

[11] With the exception of consecutive migrations from host to host that add additional communication overhead.

migrations aggregated over runtime. The baseline shows the optimal case of 600 messages consumed for parent to host migrations and host to parent returns. The baseline means that no consecutive migrations need to be performed and no corrective agents are lost in migrations. Results show that parent to host migrations are 13.69% higher than baseline, whereas host to parent returns are 3.42% lower than baseline.

Figure 4g and 4h show the rate of migration success that remains on average at high values for all departure steps in both UP-DOWN and DOWN-UP. Figure 5b summarizes the rate of migration success for varying departure steps. As the departure step increases, the rate of migration success decreases on average, suggesting that self-corrective actions are more effective when network adjustments are performed incrementally at several small steps. This is also confirmed by the number of return messages that are closer to the baseline in Figure 6c. The rate of migration success increases on average 5.57% as the departure steps changes from DS-300 to DS-75.

C. *Heavyweight network adjustments*

Figure 7 illustrates the average accuracy of the self-corrective model on DIAS under heavyweight network adjustments. The following key observations are made: (i) The departure period does not influence significantly the accuracy. (ii) In SUMMATION, the average relative error decreases 21.17% on average as the departure step changes from DS-300 to DS-50. In MAXIMUM, the respective decrease is 48.01%. (iii) In AVERAGE, no significant differences are observed. The accuracy of SUMMATION over runtime is illustrated in Figure 8c and 8d.

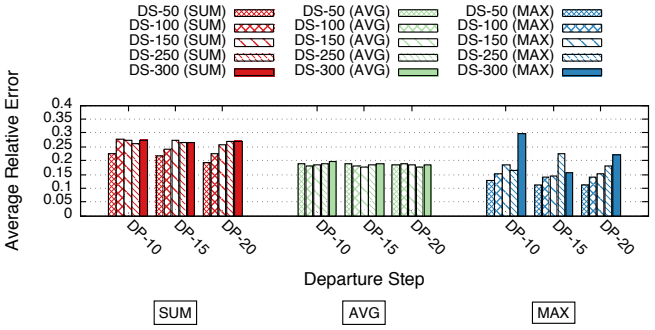
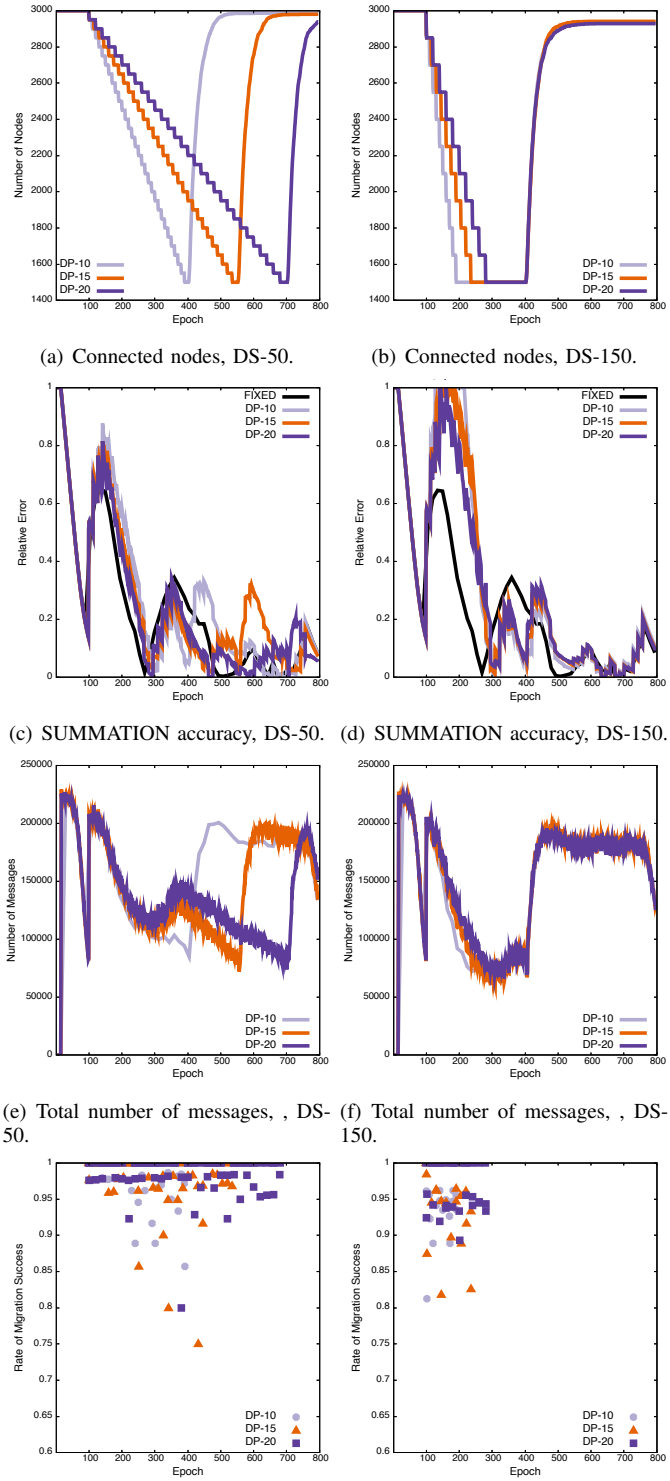


Figure 7. Average accuracy of the aggregation functions under heavyweight network adjustments.

Figure 8e and 8f illustrate the influence of the departure period on communication cost. As shown in Figure 8e for DS-50, nodes leave the network till the 700th epoch in DP-20, in contrast to DP-10 that completes the network adjustment in almost 400 epochs. The number of messages exchanged decreases when the duration of the network adjustment is longer as also confirmed in Figure 9a that shows 15.52% decrease from DP-10 to DP-20 for DS-50.

Figure 9a illustrates the aggregated total number of messages for varying departure steps and periods. The communication cost for different departure periods remains approximately



(a) Connected nodes, DS-50. (b) Connected nodes, DS-150.

(c) SUMMATION accuracy, DS-50. (d) SUMMATION accuracy, DS-150.

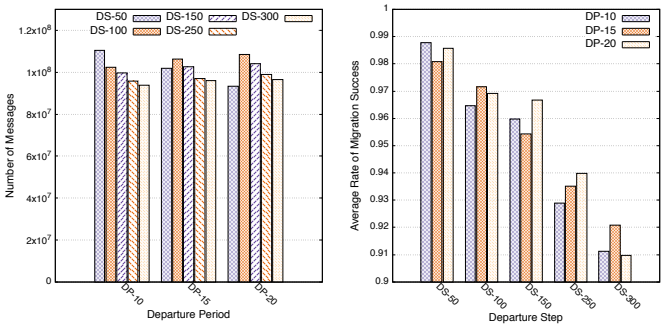
(e) Total number of messages, , DS-50. (f) Total number of messages, , DS-150.

(g) Rate of migration success, DS-50. (h) Rate of migration success, DS-150.

Figure 8. Performance of the self-corrective model on DIAS over its runtime under heavyweight network adjustments. (a)-(b) Number of nodes in the network. (c)-(d) Accuracy of SUMMATION. (e)-(f) Total number of messages. (g)-(h) Rate of migration success.

the same, however, as in the case of the lightweight network adjustment, higher departure steps have a lower communica-

tion cost, for instance, DS-300 has on average 6.29% lower number of messages than DS-50.



(a) Total number of messages. (b) Rate of migration success.

Figure 9. Total number of messages and rate of migration success aggregated over runtime under heavyweight network adjustments.

Figure 10 illustrates the number of messages originated from migrations aggregated over runtime. The baseline shows the optimal case of 1500 messages consumed for parent to host migrations and host to parent returns. As in the lightweight scenario of network adjustments, the baseline means that no consecutive migrations need to be performed and no corrective agents are lost in migrations. Results show that parent to host migrations are 26.59% higher than the baseline, whereas host to parent returns are 5.51% lower than the baseline.

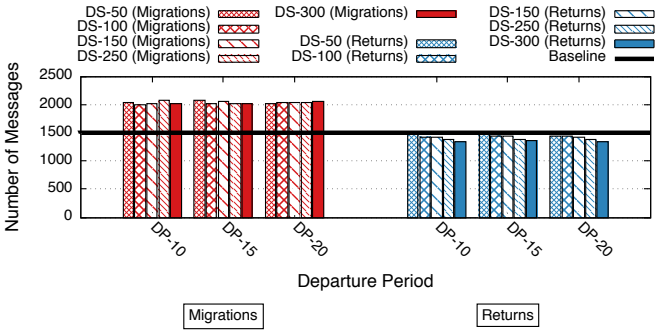


Figure 10. Number of messages aggregated over runtime originated from migrations under heavyweight network adjustments.

Figure 8g and 8h show the rate of migration success for different departure steps and periods. Strikingly, the rate remains over 70% in all cases even under heavyweight network adjustments, showing that the corrective agents manage to find a host and perform their operations even when half of the network fails. Figure 9b confirms the high average rates of migration success that increases for lower departure steps as also in the case of lightweight network adjustments. DS-50 has 7.2% higher rate of migration success than DS-300.

VI. Comparison with Related Work

To the best of authors' knowledge, there is no other corrective model for large-scale decentralized dynamic networks for a fair and meaningful quantitative comparison. Therefore, this section focuses on a qualitative comparison.

A majority of fault-detection, replication and fault-tolerance mechanisms for multi-agent systems are managed in a centralized fashion by dedicated, for this purpose, replication servers, proxies and storage of checkpoints [18], [19], [8]. In contrast, the agents of the proposed self-corrective model rely on the peer sampling service for a fully decentralized detection and orchestration of reverse computations. No additional resources are required as the corrective agents operate in a symbiotic fashion within the nodes of the network.

A self-healing web of things agent-based architecture is earlier introduced [30] that models detection, diagnosis and recovery processes. The architecture is mainly conceptual and is not experimentally validated. It focuses on healing aspects rather than corrective computations. It is not clear how it can be applied in large-scale decentralized computational networks, in contrast to this work that extensively illustrates experimental findings on self-corrective aggregation networks under challenging network scenarios of leaving and rejoining.

There is a class of fault-tolerance algorithms for computational problems with large numbers of parallel processes. Such algorithms are based on checkpoint rollback-recovery and restart mechanisms [31]. An earlier algorithmic checkpoint-free fault-tolerance approach is introduced to significantly decrease the high memory overhead of periodic checkpoints for parallel matrix computations [32]. Hybrid methods combining checkpoints and checksum storage are designed to cope with a broader range of dense matrix factorization problems [33]. In contrast to all these methods designed for failures of parallel computational processes, the proposed self-corrective model focuses on large-scale decentralized networks in which nodes may temporarily leave, fail or rejoin and do not share a common memory space or local data transfers.

Other earlier work argues that the tremendous growth in the volume of control data and application measurements, such as mobile cellular networks, often turns self-healing and fault-tolerance infeasible and unscalable without the employment of big data models and technologies [12]. In contrast, this work shows that accurate network analytics in dynamic networks can be performed out of the context of big data based on autonomic self-corrective mechanisms, i.e. the realization of the proposed self-corrective model in DIAS.

Finally, there are significant links between privacy and self-corrective computations under network adjustments. For instance, consider the computation of aggregation functions using differential privacy that is relevant in the context of reverse computations and DIAS as well. Even in the case of a single node failure, the noises in the input values are not canceled out and therefore aggregations functions cannot be computed. Grouping of nodes and data structures such as binary interval trees can make the privacy-preservation of aggregation more resilient to node failure [34], [35].

VII. Conclusion and Future Work

This paper concludes that designing large-scale dynamic computational networks with autonomic self-corrective capabilities enabled by decentralized reverse computations is

feasible. This is experimentally evaluated using real-world data from a smart grid pilot project under extreme network adjustments corresponding to catastrophic events with up to 50% of the nodes leaving the network. In contrast to related work, the proposed model does not employ additional computational resources for redundancy. Instead it utilizes the inner network resources in a symbiotic autonomic fashion to collectively orchestrate reverse computations that enable a network to be self-corrective by-design. The model proves to be generic and modular when applied in the DIAS aggregation system as no major structural changes are required in the design.

Future work includes the experimental evaluation of the self-corrective model under node failures. Expanding the scope of the model to other network dynamics, for instance, reversing computations of malicious nodes infected during system runtime can provide new insights for self-defense mechanisms in large-scale computational networks [36], [23].

REFERENCES

[1] A. Loukas, M. Zuniga, I. Protonotarios, and J. Gao, "How to identify global trends from local decisions? event region detection on mobile networks," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 1177–1185.

[2] E. Pournaras, M. Vasirani, R. E. Kooij, and K. Aberer, "Decentralized planning of energy demand for the management of robustness and discomfort," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2280–2289, 2014.

[3] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson, "Virtual trip lines for distributed privacy-preserving traffic monitoring," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 15–28.

[4] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "A self organizing wireless sensor network," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 37. The University; 1998, 1999, pp. 1201–1210.

[5] A. Guerrieri, A. Montresor, and Y. Velegrakis, "Top-k item identification on dynamic and distributed datasets," in *European Conference on Parallel Processing*. Springer, 2014, pp. 270–281.

[6] 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.

[7] 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*.

[8] R. Stanković, M. Štula, and J. Maras, "Evaluating fault tolerance approaches in multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 31, no. 1, pp. 151–177, 2017.

[9] S. Gürses, C. Troncoso, and C. Diaz, "Engineering privacy by design," *Computers, Privacy & Data Protection*, vol. 14, no. 3, 2011.

[10] P. Tambe, "Big data investment, skills, and firm value," *Management Science*, vol. 60, no. 6, pp. 1452–1469, 2014.

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

[12] E. J. Khatib, R. Barco, P. Munoz, I. D. L. Bandera, and I. Serrano, "Self-healing in mobile networks with big data," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 114–120, January 2016.

[13] 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.

[14] C. D. Carothers, K. S. Perumalla, and R. M. Fujimoto, "Efficient optimistic parallel simulations using reverse computation," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 9, no. 3, pp. 224–253, 1999.

[15] K. S. Perumalla and A. J. Park, "Reverse computation for rollback-based fault tolerance in large parallel systems," *Cluster Computing*, vol. 17, no. 2, pp. 303–313, 2014.

[16] E. Deelman, C. Carothers, A. Mandal, B. Tierney, J. S. Vetter, I. Baldin, C. Castillo, G. Juve, D. Król, V. Lynch *et al.*, "Panorama: An approach to performance modeling and diagnosis of extreme-scale workflows," *International Journal of High Performance Computing Applications*, p. 1094342015594515, 2015.

[17] D. Zhao, N. Liu, D. Kimpe, R. Ross, X.-H. Sun, and I. Raicu, "Towards exploring data-intensive scientific applications at extreme scales through systems and simulations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1824–1837, 2016.

[18] A. Fedoruk and R. Deters, "Improving fault-tolerance by replicating agents," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*. ACM, 2002, pp. 737–744.

[19] R. Singh and M. Dave, "Using host criticalities for fault tolerance in mobile agent systems," in *Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on*. IEEE, 2012, pp. 67–72.

[20] E. Pournaras, M. Warnier, and F. M. Brazier, "A generic and adaptive aggregation service for large-scale decentralized networks," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, p. 1, 2013.

[21] E. Gelenbe and Y. Wang, "A trade-off between agility and resilience," in *Proceedings of the 13th Turkish symposium on artificial intelligence and neural networks*, 2004, pp. 209–217.

[22] 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.

[23] S. Bijani and D. Robertson, "A review of attacks and security approaches in open multi-agent systems," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 607–636, 2014.

[24] R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding availability," in *International Workshop on Peer-to-Peer Systems*. Springer, 2003, pp. 256–267.

[25] S. Herker, W. Kiess, X. An, and A. Kirstädter, "On the trade-off between cost and availability of virtual networks," in *Networking Conference, 2014 IFIP*. IEEE, 2014, pp. 1–9.

[26] M. Randles, D. Lamb, E. Odat, and A. Taleb-Bendiab, "Distributed redundancy and robustness in complex systems," *Journal of Computer and System Sciences*, vol. 77, no. 2, pp. 293–304, 2011.

[27] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Redundancy, diversity, and connectivity to achieve multilevel network resilience, survivability, and disruption tolerance invited paper," *Telecommunication Systems*, vol. 56, no. 1, pp. 17–31, 2014.

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

[29] 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.

[30] R. A. Arocha, M. Manouvrier, and M. Rukoz, "An agent architecture to enable self-healing and context-aware web of things applications," in *International Conference on Internet of Things and Big Data (IoTBD 2016)*, 2016, pp. 82–87.

[31] I. P. Egwutuoha, D. Levy, B. Selic, and S. Chen, "A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems," *The Journal of Supercomputing*, vol. 65, no. 3, pp. 1302–1326, 2013.

[32] Z. Chen, "Scalable techniques for fault tolerant high performance computing," Ph.D. dissertation, University of Tennessee, 2006.

[33] A. Bouteiller, T. Herault, G. Bosilca, P. Du, and J. Dongarra, "Algorithm-based fault tolerance for dense matrix factorizations, multiple failures and accuracy," *ACM Trans. Parallel Comput.*, vol. 1, no. 2, pp. 10:1–10:28, Feb. 2015.

[34] J. Won, C. Y. Ma, D. K. Yau, and N. S. Rao, "Proactive fault-tolerant aggregation protocol for privacy-assured smart metering," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 2804–2812.

[35] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *International Conference on Financial Cryptography and Data Security*. Springer, 2012, pp. 200–214.

[36] B. Sun, X. Shan, K. Wu, and Y. Xiao, "Anomaly detection based secure in-network aggregation for wireless sensor networks," *IEEE Systems Journal*, vol. 7, no. 1, pp. 13–25, 2013.