Resilient Distributed Collection Through Information Speed Thresholds

Giorgio Audrito, Sergio Bergamini, Ferruccio Damiani, Mirko Viroli

Resilient Distributed Collection through Information Speed Thresholds

Giorgio Audrito[1][0000−0002−2319−0375], Sergio Bergamini[1], Ferruccio Damiani[1][0000−0001−8109−1706], and Mirko Viroli[2][0000−0003−2702−5702]

[1] Dipartimento di Informatica, University of Torino, Torino, Italy
{giorgio.audrito, ferruccio.damiani}@unito.it,
sergio.bergamini@edu.unito.it
[2] Alma Mater Studiorum–Università di Bologna, Italy
mirko.viroli@unibo.it

Abstract. One of the key coordination problems in physically-deployed distributed systems, such as mobile robots, wireless sensor networks, and IoT systems in general, is to provide notions of "distributed sensing" achieved by the strict, continuous cooperation and interaction among individual devices. An archetypal operation of distributed sensing is data summarisation over a region of space, by which several higher-level problems can be addressed: counting items, measuring space, averaging environmental values, and so on. A typical coordination strategy to perform data summarisation in a peer-to-peer scenario, where devices can communicate only with a neighbourhood, is to progressively accumulate information towards one or more collector devices, though this typically exhibits problems of reactivity and fragility, especially in scenarios featuring high mobility. In this paper, we propose coordination strategies for data summarisation involving both idempotent and arithmetic aggregation operators, with the idea of controlling the minimum information propagation speed, so as to improve the reactivity to input changes. Given suitable assumptions on the network model, and under the restriction of no data loss, these algorithms achieve optimal reactivity. By empirical evaluation via simulation, accounting for various sources of volatility, and comparing to other existing implementations of data summarisation algorithms, we show that our algorithms are able to retain adequate accuracy even in high-variability scenarios where all other algorithms are significantly diverging from correct estimations.

Keywords: Data Aggregation · Adaptive Algorithm · Aggregate Programming · Computational Field · Gradient

1 Introduction

Nowadays physical environments are more and more filled with heterogeneous connected devices (intelligent and mobile, such as smartphones, drones, robots). These contexts increasingly call for new mechanisms of collective adaptation, ultimately supporting a view of environments as acting as true *pervasive computing fabric*, where sensing, actuation and computation are naturally seen as

inherently resilient and distributed across physical space [16]. In this paper we are concerned with the design of a self-adaptive coordination strategy able to realise *distributed sensing* concerning physical properties of the environment or virtual/digital characteristic of the computational one. By the strict cooperation and interaction of dynamic sets of mobile entities situated in physical proximity, distributed sensing can generally support forms of complex situation recognition [18], better monitoring of physical environment [16], and observation (and then control) of teams of agents [33]. In the context of coordination models and languages, field-based coordination [23,31,32] has been recently proposed as framework to program increasingly complex self-organising coordination strategies for such scenarios.

A paradigmatic coordination operation of distributed sensing is data summarisation performed on devices filling a region of space: it is a key component on top of which one can then realise other operations such as counting, integration, averaging, maximisation, and the like. In fact, data summarisation corresponds to the *reduce* phase of the MapReduce paradigm [19] ported into a "spatial" context of agents spread in a physical environment and communicating by proximity, and has close analogues designed for wireless sensor networks [29]. Data summarisation can be solved by an algorithm of *distributed collection*, where information propagates towards one or more collector devices, and combine *en-route* until reaching a unique value, i.e, the result of collection. This component of self-organising behaviour (sometimes named the "C" building block, in short [30]), is one of the most basic and widely used components of collective adaptive systems (CASs). Seen in terms of field-based coordination, collection is essentially a distributed coordination algorithm that computes a specific case of "computational field" [3,11], namely, a data structure distributed across space such that each device holds only the local value—which, in the case of collection represents a partial result of counting in a whole sub-region. This "brick" can be applied to a variety of different contexts, as it can be instantiated for values of any data type with an associative and commutative aggregation operator.

However, implementing C can be very tricky, especially in mobile and faulty environments (i.e., with changes in the network of computational devices), which are the norm in several emerging application contexts, including airborne sensing by drones [15], crowd management by people smartphones [14], and vehicular networks [25]: existing implementations based on heuristic reasoning (single-path and multi-path [5,30]) tend to be very fragile in practice.

In this paper we present two new algorithms for effectively and efficiently carrying on the computation of the C building block, based on a theoretical approach backed up by simulation results, which is able to achieve adequate accuracy in highly volatile scenarios. In the algorithm for idempotent aggregation (e.g. set union, maximum), as for existing multi-path collection algorithms, data chunks flow through agents through many possible links of the underlying proximity network. Which links to use are selected by imposing differentiated thresholds on minimum information propagation speed, threshold which in turn are set to the highest value ensuring that data is not discarded by all neighbours

(under suitable assumptions on the network configuration). Instead, in the algorithm for arithmetic aggregation (e.g. sum, product), data chunks flow through a single outgoing link selected to ensure the maximum information propagation speed in the worst-case scenario. In both arithmetic and idempotent aggregation, the algorithms chosen are designed to maximise the worst-case information propagation speed under the given assumptions. Notice that which of the two algorithms applies depends only on the problem at hand and not on the runtime setup of a network. Thus, a system designer can decide which of the two algorithms are to be exploited depending on the properties of the aggregation operator only, and there is no overlap: arithmetic operators are never idempotent.

We validate the performance of the algorithms in archetypal situations, taking into account agent mobility and discontinuities in network configuration, as well as network size and density. Ultimately, by accounting for various sources of volatility, using different state-of-the-art distance estimations, and comparing to other existing implementations of aggregation algorithms, we show that these algorithms are able to retain acceptable precision even in high-variability scenarios where all other algorithms are significantly diverging from correct estimations.

The work of this paper is arguably a significant step in the context of engineering CASs. In general, the proposed coordination algorithm can be used as a solid component for engineering collection services in highly distributed and mobile systems. On the other hand, in the specific context of field-based coordination and aggregate computing framework [14], these algorithms provide an implementation for the fundamental "C block" as advocated in [30], coupling that of "G block" as of [6], and together forming a set of combinators effectively supporting construction of higher-level, self-stabilising coordination strategies in mobile distributed systems, such as e.g. the SCR pattern proposed in [17].

The remainder of this paper is organised as follows. Section 2 presents the state-of-the-art in data summarisation techniques and necessary backgrounds. Section 3 presents the algorithms together with the assumptions that ensure achieving optimal reactivity. Section 4 compares these algorithms with the state-of-the-art in archetypal scenarios particularly hard for summarising algorithms. Finally, Section 5 concludes with directions of future research.

2 Background and Related Work

2.1 Computational Model

In aggregate programming [14], a distributed network consists of mobile devices, capable to perform asynchronous computations and interacting by exchanging messages. Every device performs periodically the same sequence of operations, with an usually steady rate T: collection of received messages, computation, and transmission of messages. The instants and places when and where devices start their computation are called *events* ϵ, and constitute basic element modelling the system evolution. Every event is a spatio-temporal point, happening on a

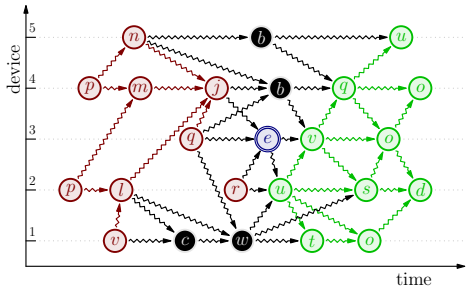

Fig. 1. Representation of an event structure, together with literal values depending on events. Past events of event e (circled blue) are depicted in red, future events in green, concurrent events in black.

device $\delta(\epsilon)$ at a certain moment in time $t(\epsilon)$ and position in space $\boldsymbol{p}(\epsilon)$. The values manipulated by an aggregate program are distributed in space and evolve in time, and can thus be represented as functions of events $v(\epsilon)$. Furthermore, events are structured by the message-passing relation among them.

Definition 1 (neighbour). *An event ϵ' is a* neighbour *of an event ϵ, denoted as $\epsilon' \rightsquigarrow \epsilon$, if a message sent by ϵ' was the last from $\delta(\epsilon')$ able to reach $\delta(\epsilon)$ before ϵ occurred (and has not been discarded as obsolete since).*

Note that, in an actual asynchronous distributed system, a device could fire more frequently than another, hence multiple messages from a "fast" device could reach a "slow" target before it can fire a new round: the above definition will allow us to focus only on the latter received one. Similarly, no messages from a "slow" device could reach a "fast" target during a round, and the above definition allows to retain messages from such a slow device across rounds, increasing the computation stability. Details on when messages are persisted or discarded are not given in the definition, leaving them as a choice during system design.

The neighbouring relation on events forms a direct acyclic graph (DAG), since it is time-driven and anti-symmetric (unlike spatial-only neighbouring which is usually symmetric). The transitive closure of this relation defines the *causality* partial order $\leq$, so that $\epsilon' \leq \epsilon$ iff there exists a sequence of events $\epsilon' \rightsquigarrow \ldots \rightsquigarrow \epsilon$ connecting ϵ' to ϵ. The causality relation defines which events constitute the past, future or are concurrent to any given event. A set of events with a neighbouring and causality relation is also called *event structure*[3] (represented in Figure 1), and provides a basis to formally define the behaviour of a distributed system. In the remainder of this paper, we shall use the following quantities and primitives:

– the radius R within which communication succeeds;[4]

[3] Event structures for Petri Nets are used to model a spectrum of *possible evolutions* of a system, hence include also an *incompatibility* relation, discriminating between alternate future histories and modelling non-deterministic choice. However, following [21], we use event structures to model a "timeless" *unitary history* of events, thus avoiding the need for an incompatibility relation.

[4] In reality, the communication range of a node is very irregular. As suggested by Zhou et al. [35], such an irregular radius can be bounded, justifying the usage of a fixed quantity.

- the device $\delta(\epsilon)$ and time $t(\epsilon)$ in which event ϵ takes place;
- the time difference (lag) between neighbour events $\mathrm{lag}(\epsilon', \epsilon) = t(\epsilon) - t(\epsilon)$;[5]
- the measured distance between neighbour events $\mathrm{dist}(\epsilon', \epsilon)$, possibly affected by errors.

The latter can be obtained in three main different ways, depending on the time to which the two positions $\boldsymbol{p}'$ and $\boldsymbol{p}$ involved refer to: *(i)* in GPS-based systems, $\boldsymbol{p}'$ is the position measured in $t(\epsilon')$ and $\boldsymbol{p}$ is the position measured in $t(\epsilon)$; *(ii)* if distance is sensed at message receival, both positions refer to $t(\epsilon')$; *(iii)* if distance can be sensed in every moment, then both positions may refer to $t(\epsilon)$.

Throughout the description of algorithms we will use the notation $X(\epsilon)$ to represent a distributed value X depending on *events*, while $X_{\epsilon'}(\epsilon)$ will symbolize a value depending on *neighbouring relationships* $\epsilon' \rightsquigarrow \epsilon$, that is, a quantity computed in ϵ with respect to a neighbour event ϵ'.

2.2 Self-Stabilising Building Blocks

Recent works promoted an approach to engineer complex field-based coordination algorithms by combination of basic building blocks [30], capturing key mechanisms of self-organisation such as spreading (block "G"), collection (block "C"), time evolution (block "T"), leader election and partitioning (block "S"), measuring centrality [7] and so on. For instance, self-organising coordination regions can be developed by a S-G-C-G composition [17].

The most basic and versatile building block is called *gradient* (G block), which provides distance estimation, creating a spanning tree and performing broadcast operations. In particular, the *potential field* $P(\epsilon)$ of distances from a source is a crucial input of every data aggregation routine (C block), providing means to guide the direction of aggregation. Accurately computing distances in a distributed and volatile scenario is a demanding task, which can be tackled in different ways depending on the context. In spite of variations, the general framework is that of gradient-based *field computations* [23,24], where local estimates from the source are repetitively shared with neighbours and combined with proximity estimates of mutual distance.

If no proximity sensors are available, the harsh *hop-count* measure can be improved through statistical tools [22], obtaining continuous and adaptive distance estimates. Furthermore, even when a proximity sensor is available, reactivity to input changes and network variability may be impaired by the *rising value problem*[6]—simply, reaction to changes causing increase of distance is very low [9]. Several solutions have been proposed to tackle this problem. Following recent reviews of distance estimation algorithms [6,9] three solutions are shown to always outperform basic algorithms: FLEX [12], BIS [8], and ULT [6].

FLEX is an algorithm aimed at maximising stability of values while containing the error within predictable bounds, which also addresses the rising value

[5] Note that this quantity can be computed with reasonable accuracy even in absence of a global clock [10].

[6] Also known as the *count to infinity* problem in routing algorithms.

problem by introducing a metric distortion. BIS, instead, exploits time information in order to solve the rising value problem obtaining optimal single-path reactivity to input changes, without concerns on value stability. ULT develops on BIS by adding a stale values detector running at (faster) multi-path speed, while addressing value stability with the addition of filters and dampers. Being obtained by the integration of different methods, ULT is tuned by a large number of parameters, and can range to being almost identical to BIS (when filters and dampers are disabled) to being closer to FLEX (when dampers are active).

2.3 Distributed Data Collection

Data collection (also called aggregation) is a key component of distributed algorithms. It has been tacked in different ways depending on the application context (like, e.g., wireless sensor networks [26,29], high-performance computing [19] and spatial computing [13]). Notably, all of these different approaches rely on the same basic mechanisms. In data collection, distributed values are combined together through an aggregation operator $\oplus$ that enjoys the following properties:

1. *commutativity*: $u \oplus v = v \oplus u$;
2. *associativity*: $u \oplus (v \oplus w) = (u \oplus v) \oplus w$.

Provided that the above properties hold, the aggregation $\bigoplus \mathcal{C}$ of the elements of a multi-set $\mathcal{C}$ is well-defined (the order in which the individual elements are aggregated is immaterial). Some common aggregation operators are the idempotent operators maximum and minimum, and the arithmetic operators addition and multiplication. Scenarios with intrinsic communication errors and input volatility (like, e.g., wireless sensor networks and spatial computing) require to consider a further property:

3. *continuity*: the effect on the aggregation of a certain percentage p of errors tends to zero as p tends to zero.

This property holds for the idempotent and arithmetic aggregation operators cited above, however, it does not hold for other operations like, e.g., modular sum: the modular addition of a single spurious element can fully disrupt the outcome of the aggregation of an arbitrary big collection of elements.

In the context of an environment with proximity-based interactions, given a commutative and associative operator, a data aggregation algorithm asynchronously combines input values $x(\epsilon)$ from different devices into a single value in a selected device called *source* (or *collector*). The algorithm manages the flow of data towards the source to avoid multiple aggregation of the same values. This twofold prerequisite, of *acyclic* flows *directed* towards the source, is met by relying on a given *potential field* $P(\epsilon)$, approximating a certain measure of distance from the selected source. As long as information flows descending the potential field, cyclic dependencies are prevented and eventual reaching of the source is guaranteed. For each event ϵ, potential descent is enforced by splitting

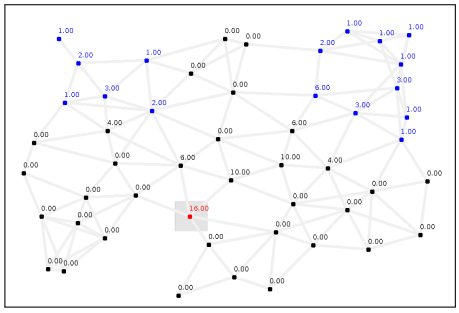

Fig. 2. A collection field in a p2p scenario that, by using single-path aggregation, counts the number of blue agents and collects the result in the red agent. Each agent holds a partial result of counting, based on how many "single-path flows" from blue agents to red agent cross it. Connections are bidirectional, and aggregation flows from smaller to greater values.

the set of neighbours events $E_\epsilon = \{\epsilon' \mid \epsilon' \rightsquigarrow \epsilon\}$ according to their potential value into the two disjoint sets:

$$E_\epsilon^- = \{\epsilon' \in E_\epsilon \mid P_{\epsilon'}(\epsilon) < P(\epsilon)\} \qquad \text{and} \qquad E_\epsilon^+ = \{\epsilon' \in E_\epsilon \mid P_{\epsilon'}(\epsilon) > P(\epsilon)\}\,.$$

Thus, values can be received only from E_ϵ^+ and must be sent only to E_ϵ^-. Three main algorithms implementing the collection block have been proposed so far: *single-path*, *multi-path* and *weighted multi-path*, all scaling to arbitrarily large systems as they require constant computational resources per node.

Single-path Aggregation. The single-path algorithm C_{sp} ensures that information flows through a forest in the network, by sending the whole partial aggregate $C_{\text{sp}}(\epsilon)$ computed during event ϵ to the single neighbour $m(\epsilon) = \epsilon'$ with minimum potential $P_{\epsilon'}$ among all neighbour events in E_ϵ. This is accomplished by repeatedly applying the following rule:

$$C_{\text{sp}}(\epsilon) = x(\epsilon) \oplus \bigoplus_{\epsilon' \in E_\epsilon^+ \,\wedge\, \delta(m(\epsilon')) = \delta(\epsilon)} C_{\text{sp}}(\epsilon') \tag{1}$$

Equation 1 computes the partial aggregate in ϵ by combining together the local input value $x(\epsilon)$ and the partial aggregates from direct predecessors ϵ' with higher potential for which $\delta(\epsilon)$ is the selected device $\delta(m(\epsilon'))$. A screenshot of this algorithm after convergence is reached is shown in Figure 2.

Since data flows descending the potential as fast as possible, single-path aggregation attains optimal reactivity to input changes in static environments. However, in mutable environments, the message from ϵ to $m(\epsilon)$ may be lost, disrupting communication and pruning the entire branch of the forest rooted in ϵ. This phenomenon translates into poor performances, provided that values far from the source contribute significantly to the aggregation (e.g., non-zero values for summation, high values for minimisation, and so on).

Multi-path Aggregation. The multi-path algorithm C_{mp} allows information to flow through every path compatible with the given potential field. In order

to avoid double counting, it is thus necessary to divide the partial aggregate of an event ϵ equally among *every* event ϵ' with lower potential, by iteratively applying the following rule:

$$C_{\mathrm{mp}}(\epsilon) = x(\epsilon) \oplus \bigoplus_{\epsilon' \in E_\epsilon^+} \{C_{\mathrm{mp}}(\epsilon') \oslash N(\epsilon')\} \tag{2}$$

where $N(\epsilon) = |E_\epsilon^-|$ and $\oslash$ is a binary operator such that $v \oslash n$ means "dividing by n", i. e., an element that aggregated with itself n times produces the original value v. Since information needs to be "divisible" for $\oslash$ to exist, two categories of aggregation operators are supported:

1. *arithmetic operations*, e.g., point-wise sum and multiplication of vectors $\boldsymbol{v} \in \mathbb{R}^n$ of real numbers (for which $\oslash$ is respectively division and root extraction);
2. *idempotent operations*, e.g., computation of maximum and minimum among values v in a partially ordered set (for which $\oslash$ is the identity function).

Thus, theoretically, multi-path has a narrower scope than single-path. However, the vast majority of practically occurring (continuous) aggregation operators can be typically recast to be either arithmetic or idempotent. In particular, idempotent operations have been used to emulate several different aggregations through statistical tools: distinct count, sum, uniform sampling, selection of most frequent values [26], and order statistics [34].

Since data flows through every possible path, it is unlikely for devices to be excluded from the aggregation, thus preventing data loss. On the other hand, the reactivity to input changes of multi-path aggregation is particularly poor. In fact, even in static environments, values flow through every possible path including the *longest* path, forcing reaction to changes to be delayed until all paths have been exploited (in particular for idempotent operations), and resulting in a reaction speed inversely proportional to the device density. In mutable environments, the problem is further exacerbated by the creation of *information loops*, which occur when two or more moving devices of similar potential invert their relative potential order in consecutive rounds, causing information from a device δ to come back to the same device, slowing down even further the reaction speed of the algorithm, and inducing exponential overestimations in the arithmetic case.

Weighted Multi-path Aggregation. Recent works [4,5] develop on the multi-path algorithm, by allowing partial aggregates to be divided unequally among neighbours. Weights corresponding to neighbours are calculated in order to penalise devices that are likely to lose their "receiving" status, a situation that can happen in two cases:

1. if the "receiving" device is too close to the edge of proximity of the "sending" device, so that it might step outside of it in the immediate future breaking the connection;
2. if the potential of the "receiving" device is too close to the potential of the "sending" device, so that their relative role of sender/receiver might be

switched in the immediate future, possibly creating an "information loop" between the two devices.

Both situations are addressed by a weight function $w_{\epsilon'}(\epsilon) = d(\epsilon', \epsilon) \cdot p(\epsilon', \epsilon)$, measuring how much of the information from ϵ should be sent to a neighbour $\delta(\epsilon')$ as the product of the two corresponding factors $d(\epsilon', \epsilon) = R - \text{dist}(\epsilon', \epsilon)$ and $p(\epsilon', \epsilon) = |P(\epsilon) - P(\epsilon')|$, where R is the communication radius and $\text{dist}(\epsilon', \epsilon)$ the distance measured between the events. Since these weights do not sum up to any particular value, they need to be normalised by the factor $N(\epsilon) = \sum_{\epsilon' \in E_\epsilon^-} w_{\epsilon'}(\epsilon)$, obtaining normalised weights $w_{\epsilon'}(\epsilon)/N(\epsilon')$. The partial aggregates accumulated by devices can then be calculated as in C_{mp} (see 2) with the addition of weights, by iteratively applying the following rule:

$$C_{\text{wmp}}(\epsilon) = x(\epsilon) \oplus \bigoplus_{\epsilon' \in E_\epsilon^+} \left\{ C_{\text{wmp}}(\epsilon') \otimes \frac{w_{\delta(\epsilon)}(\epsilon')}{N(\epsilon')} \right\} \tag{3}$$

where $\otimes$ is a binary operator such that $v \otimes k$ "extracts" a certain percentage k of a local value v.[7] In particular, if $\oplus$ is arithmetic (addition) then $\otimes$ is multiplication, whereas if $\oplus$ is idempotent then $\otimes$ is a threshold function regulating which links should be exploited for transmission and which should be ignored.

This algorithm has been shown to significantly outperform both the single-path and multi-path strategies, however, it is based on heuristics hence cannot provide correctness guarantees: in fact, it produces exponentially growing peaks of error for arithmetic aggregations in scenarios with high mobility [5].

3 Collection by Lossless Information Speed Thresholds

In this section, we present the *Lossless Information Speed Thresholds* collection algorithm (C_{list}). It maximises information speed under the general assumptions presented in Section 2.1 and the additional assumptions on the network model given in Section 3.1, with respect to the algorithms satisfying the constraints given in Section 3.2.

3.1 Network Model Assumptions

As for the other summarisation algorithms, we assume a potential field $P(\epsilon)$ to be available as input in each event. Given an event ϵ, we denote as ϵ_{next} the following event on the same device, so that $\epsilon \rightsquigarrow \epsilon_{\text{next}}$ and $\delta(\epsilon) = \delta(\epsilon_{\text{next}})$. In order for C_{list} to be computed, we need a minimal degree of forecasting values in next events ϵ_{next}, as stated by the following assumptions.

- *Sure connection.* For each event ϵ and neighbour ϵ', there is a Boolean value $\text{surelyConnected}_{\epsilon'}(\epsilon)$ which is true iff ϵ is sure that its messages will be received by the next event ϵ'_{next} on $\delta(\epsilon')$, and is true for at least one neighbour

[7] We also used the notation $w_\delta(\epsilon')$ as alias of $w_{\epsilon''}(\epsilon')$ where $\delta(\epsilon'') = \delta$.

event ϵ'. Such value can be computed using an upper bound on distance $\text{dist}(\epsilon', \epsilon)$ together with a lower bound on connection radius R and possibly an upper bound V on device movement speed, as in the following:

$$\text{maxDistNow}(\epsilon', \epsilon) = \text{dist}(\epsilon', \epsilon) + kV \,\text{lag}(\epsilon', \epsilon) \tag{4}$$

$$\text{surelyConnected}_{\epsilon'}(\epsilon) \Leftrightarrow \text{maxDistNow}(\epsilon', \epsilon) \leq R \tag{5}$$

where k is 0 if dist refers to $t(\epsilon)$, 1 if it refers to both $t(\epsilon')$ and $t(\epsilon)$ (GPS-based), 2 if it refers to $t(\epsilon')$ (see Section 2.1).

- *Scheduled time.* For each event ϵ, we assume that an upper bound $t^u(\epsilon)$ to $t(\epsilon_{\text{next}})$ is known. Notice that this is easily satisfied with high accuracy, as activations need to be scheduled and do not happen randomly.
- *Potential evolution.* For each event ϵ, we assume that an upper bound $P^u(\epsilon)$ to $P(\epsilon_{\text{next}})$ is known. For instance, given the upper bound V on device movement speed, we may set $P^u(\epsilon) = P(\epsilon) + V \cdot (t^u(\epsilon) - t(\epsilon))$. This bound may need to be corrected for the error on potential computations, and could be significantly improved if the movement direction is known.

3.2 Algorithmic Constraints

Under the previous assumptions, we focus on collection algorithms satisfying the following constraints.

- *Lossless.* A collection algorithm is *lossless* if it ensures that the input value $x(\epsilon)$ in any event participates in the outcome $C(\epsilon')$ of the algorithm for at least one event ϵ' on the collection source (that is, such that $P(\epsilon') = 0$).
- *Scalable.* We say that a distributed algorithm is *scalable* if it uses $O(1)$ message size and $O(N)$ computation time and space in every event ϵ, where N is the number of neighbours $N = |E_\epsilon|$.

3.3 Idempotent Aggregation

In the idempotent case data duplication is not an issue, and thus data loss can be easily avoided by resorting to a multi-path algorithm. However, as we will see in Section 4.1, plain multi-path is slow in recovering to the point of being effectively equivalent to a gossip algorithm [20]. We thus propose an algorithm that adopts intermediate strategy (as in previous heuristic-based attempts [4,5]), which transmits data on a selected set of links, maximising the *speed of information flow* v (measured as units of potential descended over time) under the assumptions on the network model illustrated in Section 3.1. In fact, by discarding for every starting event ϵ the longer paths towards the source and preserving the shortest ones, we ensure that old information is quickly discarded, thus allowing the algorithm to promptly adjust to input changes.

Notice that it is not possible for a *scalable* algorithm to select paths for their overall information speed v, since partial results would not be locally computable in intermediate events. Given the candidate values i reaching a same event with

a potential descended of ΔP_i and a time elapsed of Δt_i, we need to select a constant-sized subset of them, without knowing the additional time Δt needed to reach the source, and thus the overall speed that each candidate may achieve. Thus, we indirectly select paths by imposing speed constraints in *each one* of their edges.

Given a potential field $P(\epsilon)$ of distances from the source, we compute a *threshold speed* $\theta(\epsilon)$ for each event ϵ, so that a message $\epsilon \rightsquigarrow \epsilon'$ is discarded iff:

$$v(\epsilon, \epsilon') = \frac{P(\epsilon) - P(\epsilon')}{t(\epsilon') - t(\epsilon)} < \theta(\epsilon) \tag{6}$$

that is, the information from ϵ to ϵ' is descending the potential at a speed lower than the threshold $\theta(\epsilon)$ computed in ϵ. We allow these thresholds to depend on the event, as a fixed global threshold can easily induce loss of data for large parts of the network. Furthermore, we compute these thresholds as the maximal (in order to prune the most paths possible) granting that at least one neighbour will not discard the message (*lossless* algorithm).

In order to compute these thresholds efficiently and effectively, we base on the network model assumptions in Section 3.1. For each event ϵ, we need to prevent at least one of the neighbour events $\epsilon' \rightsquigarrow \epsilon$ for which $\text{surelyConnected}_{\epsilon'}(\epsilon)$ is true from discarding the message. We then use $P^u(\cdot)$ and $t^u(\cdot)$ to predict a lower bound on the speed of the information flowing from ϵ to ϵ'_{next}:

$$v(\epsilon, \epsilon'_{\text{next}}) = \frac{P(\epsilon) - P(\epsilon'_{\text{next}})}{t(\epsilon'_{\text{next}}) - t(\epsilon)} \geq \frac{P(\epsilon) - P^u(\epsilon')}{t^u(\epsilon') - t(\epsilon)} = v^{\text{wst}}_{\epsilon'}(\epsilon) \tag{7}$$

Thus, the maximum threshold ensuring no data loss is the following:[8]

$$\theta(\epsilon) = \max\left\{ v^{\text{wst}}_{\epsilon'}(\epsilon) : \ \text{surelyConnected}_{\epsilon'}(\epsilon) = \top \right\} \tag{8}$$

The partial aggregates accumulated by devices can then be calculated by iteratively applying the following rule:

$$C_{\text{list}}(\epsilon) = x(\epsilon) \oplus \bigoplus_{\epsilon' \in E_\epsilon} \left\{ C_{\text{list}}(\epsilon') : \ v(\epsilon', \epsilon) = \frac{P(\epsilon) - P(\epsilon')}{t(\epsilon) - t(\epsilon')} \geq \theta(\epsilon') \right\} \tag{9}$$

The algorithm C_{list}, globally defined by eqs. (7) to (9), computes the partial aggregate associated with event ϵ by combining together the local value $x(\epsilon)$ and the partial aggregates from direct predecessors ϵ' for which the true information speed $v(\epsilon', \epsilon)$ was above the threshold computed in the previous events $\theta(\epsilon')$. Although every event computes the threshold by maximising the expected future information speed, and thus choosing a neighbour that theoretically guarantees the best speed, C_{list} is not a single-path algorithm: messages $\epsilon \rightsquigarrow \epsilon'_{\text{next}}$ can flow at speed greater than the estimated $v^{\text{wst}}_{\epsilon'}(\epsilon)$ (defined in Equation (7)) and thus pass the threshold even though the threshold was not designed for them.

According to the above explanation, the following property holds.

[8] If no neighbour satisfies $\text{surelyConnected}_{\epsilon'}(\epsilon)$, the no-data-loss requirement is not satisfiable and the threshold is set to $-\infty$, thus falling back to a gossip algorithm.

Property 1 (C_{list} local optimality among lossless collection algorithms). Let $\theta(\epsilon)$ be such that using information available in an event ϵ it is possible to guarantee a lowest speed of information exiting ϵ of at least $\theta(\epsilon)$ without data loss. Then the lowest speed of information exiting ϵ for C_{list} is at least $\theta(\epsilon)$.

3.4 Arithmetic Aggregation

In the arithmetic case, the situation is made more challenging by the necessity of avoiding data duplication, which can in this case lead to exponentially increasing overestimates. In order to avoid it, we modify C_{list} to become a purely single-path algorithm,[9] although the main structure remains the same. Based on eqs. (6) to (8), we choose a selected neighbour $m(\epsilon)$ maximising $v^{\text{wst}}_{m(\epsilon)}(\epsilon)$:[10]

$$m(\epsilon) \in \left\{\epsilon' \in E_\epsilon : \ \text{surelyConnected}_{\epsilon'}(\epsilon) = \top \ \wedge \ v^{\text{wst}}_{\epsilon'}(\epsilon) = \theta(\epsilon)\right\} \tag{10}$$

Partial aggregates can then be accumulated as in C_{sp} (see 1):

$$C_{\text{list}}(\epsilon) = x(\epsilon) \oplus \bigoplus_{\epsilon' \in E_\epsilon \wedge \ \delta(m(\epsilon')) = \delta(\epsilon)} C_{\text{list}}(\epsilon') \tag{11}$$

Thus, the C_{list} algorithm for arithmetic aggregation computes partial aggregates by combining together the local value $x(\epsilon)$ and the partial aggregates from direct predecessors ϵ' for which $\delta(\epsilon)$ was the selected device $\delta(m(\epsilon'))$.

4 Experimental Evaluation

We compared the new algorithm against reference single-path, multi-path and weighted multi-path implementations (*sp* [30], *mp* [30], *wmp* [5]). The algorithms were implemented in Protelis [28], which is an implementation of the *field calculus* [11] universal language for field-based computations [3]. In particular, the extension of the field calculus with the *share* operator was used [2] in order to ensure maximal efficiency.

The potential estimates guiding aggregation were computed using the state-of-the-art algorithm BIS introduced in [8] (see Section 2.2) ensuring theoretically optimal recovery speed. We also tested the usage of an exponential back-off filter to stabilise the collection results: however, we report in the following graphs only its usage for *list* on arithmetic aggregation, since it was the only case where it had a positive effect. For both the idempotent and arithmetic case, the same archetypal scenarios were selected according to the guidelines developed in [9]. The scenarios consisted of a variable number of devices with almost identical computation rate (1% systematic and accidental error) and unit disc communication model, randomly distributed in a circular area with a source device on

[9] We also need to guarantee that a message from an event ϵ is not able to reach more than one event on a same device, that is, messages are not retained across rounds.

[10] If no neighbour satisfies $\text{surelyConnected}_{\epsilon'}(\epsilon)$, the no-data-loss requirement is not satisfiable and we select the neighbour $m(\epsilon)$ minimising the probability of data loss.

the right end of the circle at simulation start, then discontinuously moved to the left end. Devices were moving at constant speed through randomly selected waypoints within the area. The scenarios were tested varying the three fundamental characteristics of such a network (all normalised in order to abstract from a specific communication radius or computation rate):

Hop diameter: the diameter of the circular area where devices are randomly displaced, measured as the number of communication radiuses (hops) contained. Values from 2 to 16 were considered (with a step of 1), using 10 when evaluating the other characteristics.

Neighbourhood size: the average number of devices in a communication radius area. Values from 5 to 40 were considered (with a step of 2.5), using 25 when evaluating the other characteristics.

Device speed: the movement speed of devices, measured as a percentage of the communication radius area covered during one computation round. Values from 0 to 50% were considered (with a step of 2.5%), using 25% when evaluating the other characteristics.

For each of the resulting 49 different scenarios, 10 runs with different random seeds were performed, averaging the results.[11] The default values (10 hops, 25 neighbours, 25% speed) were chosen after a broader search in the parameter space, as they were good representatives of the behaviour for most considered parameter values. The simulations were obtained with Alchemist as simulator [27] and the supercomputer OCCAM [1] as platform.[12]

4.1 Idempotent Aggregation

We tested collection for idempotent operators by setting $\oplus = \min$ and values to be aggregated chosen to make the aggregation as difficult as possible, showcasing every possible source of error. In fact, a difficult idempotent aggregation problem requires both *obsolete* and *distant* values to be able to significantly contribute to the aggregation. If obsolete values have a negligible impact, multi-path collection is optimal as it does not need to react to environmental changes. If distant values have a negligible impact, single-path collection is optimal since even a small coverage of the network may be sufficient.

In order to maximise the impact of distant values, we selected a set X of devices at the opposite border of the circular area with respect to the active source. Devices in X transmit a changing value which will be the result of the aggregation, while devices outside X have a fixed high value (set to 400) which is never the minimum. In order to showcase the impact of obsolete data, the values transmitted in X were changing in time according to the following sinusoidal-like function (see Figure 3 for a graphical depiction):

$$x(\epsilon) = \min(\max(A\cos(2\pi(\min(t(\epsilon), 300) + \phi)/T), -M), M)$$

[11] As the variance between the runs for arithmetic aggregation was significantly high, data was aggregated with median instead of mean.

[12] The actual code experiment is available at `https://bitbucket.org/gaudrito/experiment-optimal-collection`.

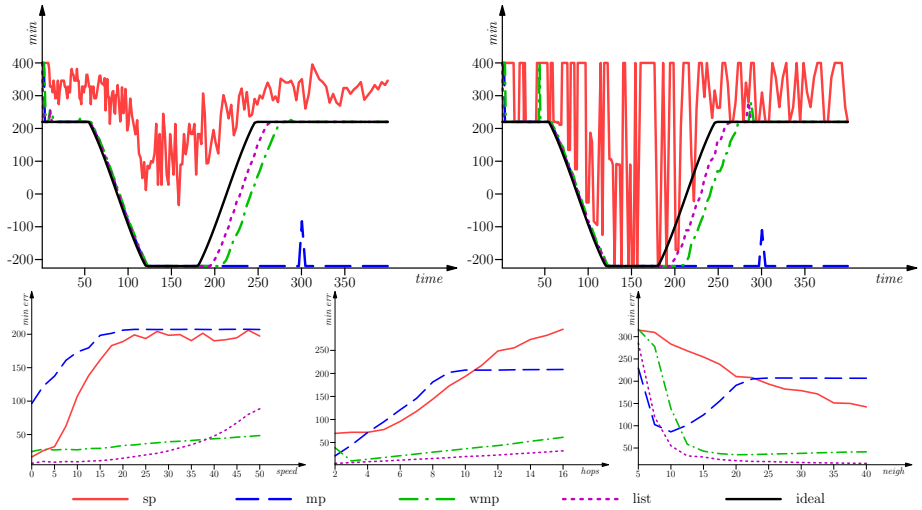

Fig. 3. Idempotent aggregation through different algorithms (sp=*single-path*, mp=*multi-path*, wmp=*weighted multi-path*, list=*lossless information speed thresholds*). Aggregation results are shown for a single run (top right) and averaged among 10 runs (top left) and $hops = 10$, $neigh = 25$, $speed = 25$. Aggregation error is shown for varying *speed*, *hops* and *neigh*, averaged among 10 runs and 400 simulated rounds (bottom).

where $t(\epsilon)$ is the time elapsed from the start of the simulation, $A = 300$ is the amplitude, $T = 250$ is the period, $\phi = -25$ is the phase, with values capped to stay within $\pm M = \pm 220$. Furthermore, at the time $t = 300$ of source switch, $x(\epsilon)$ becomes a constant equals to 220. This allows to see behaviour in all possible conditions: after a disruption, under steady inputs, and when input rises or drops.

Figure 3 summarises the evaluation results. Single-path proves to be unable to properly collect values from X in most situations except for some short time intervals, thus showing extreme variability in results, except when the number of hops is small, neighbourhood sizes are high and devices speeds are low. Multi-path produces very good results until $t = 200$, but is unable to recover when the input rises (not even after a source change), in fact behaving as a gossip algorithm, except for small networks with low density and speeds. Weighted multi-path performs quite well in all configurations, but is outperformed by *list* in all cases except for very high speeds ($> 40\%$). At such high speeds, avoiding information losses forces *list* to choose a pessimistically low threshold, that could be significantly higher while keeping a low (but non-zero) probability of loss. Finally, notice that the source switch has a minimal impact on all algorithms for idempotent aggregations.

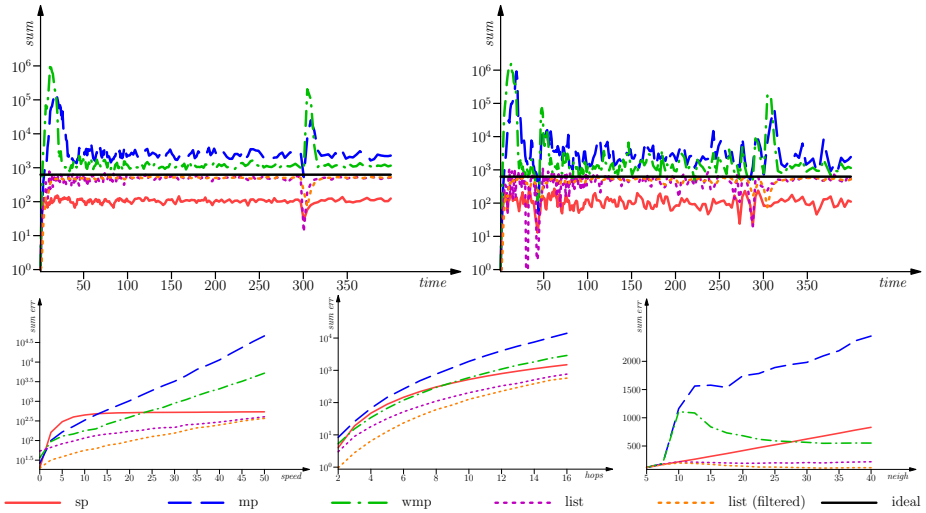

Fig. 4. Arithmetic aggregation through different algorithms (sp=*single-path*, mp=*multi-path*, wmp=*weighted multi-path*, list=*lossless information speed thresholds* with/without filter). Aggregation results are shown for a single run (top right) and averaged among 10 runs (top left) and $hops = 10$, $neigh = 25$, $speed = 25$. Aggregation error is shown for varying *speed*, *hops* and *neigh*, averaged among 10 runs and 400 simulated rounds (bottom).

4.2 Arithmetic Aggregation

We tested collection for arithmetic operators by setting $\oplus = +$ and values $x(\epsilon) = 1$ for each device. This choice amounts to *counting the total number of devices*, which is a commonly used routine and a paradigmatic example of arithmetic aggregation. We run 10 instances of each scenario and computed median results, as the relative standard errors between runs were significantly high: Figure 4 summarises the evaluation results.

The single-path (*sp*) and multi-path (*mp*) algorithms score the worst results. Single-path underestimates the ideal value by a factor of 10 at all speeds above 5%, error that gets worse as the total number of devices increases (both by *hops* or *neigh*), showing the existence of an upper bound to the number of devices that are able to reach the source. Conversely, multi-path significantly overestimates the ideal value with errors that grow approximately linearly with the number of hops or neighbours, and exponentially with speed. Weighted multi-path, shows a behaviour similar to multi-path but with a lower error: in particular, unlike *mp*, the error decreases as the number of neighbours increases, showing better performance in high density scenarios. Finally, *list* scores the best performance in every scenario, only slightly underestimating the ideal value, with an error that tends to zero as the number of neighbors increases, and is reasonably small (below 10%) even for speeds around 30%. Unlike for the other algorithms, adding an exponential back-off filter further improves the performance.

Notice that the source switch at $t = 300$ has the effect of disrupting the aggregation process for a short period of time, during which the algorithms show some positive (for multi-path based algorithms *mp*, *wmp*) or negative peaks (for single-path based algorithms *sp*, *list*). The recovery time after the switch is similar across algorithm, although the positive peaks are larger in size (overestimating the value by about 3 orders of magnitude). As shown in Figure 4 (top right), *mp* and *wmp* are always highly unstable, with peak overestimations of $5\times$; while *sp* and *list* have a more contained (while still significant) degree of instability.

5 Contributions and Future Work

In this paper, we presented two new algorithms tackling the established problem of data summarisation, both for idempotent and arithmetic operations. These algorithms are designed to maximise the speed of information flow (which translates into reactiveness to input changes) under the constraint of *no data loss*. We evaluated these algorithms in archetypal scenarios of maximal hardness, varying all fundamental (dimensionless) characteristics of a distributed network: diameter in hops, average number of neighbours, and node speed (relative to the ratio between communication radius and computation period). Overall, these algorithms significantly outperform the state-of-the-art, obtaining sound results even in scenarios with high mobility.

However, there is still some margin of future improvement. In very high mobility settings, the no-data-loss constraint forces our algorithms to an overly pessimistic behaviour, thus losing performance with respect to heuristic (lossy) techniques. In this case, future algorithms enforcing a relaxed constraint of a *maximum expected percentage* of data loss may allow for a more effective choice of the thresholds. Furthermore, our algorithms rely on a rough prediction of quantities (time and potential) across rounds: future work may directly address the prediction step, as more accurate predictions will directly translate into higher information speed thresholds, and thus reactiveness.

References

1. Aldinucci, M., Bagnasco, S., Lusso, S., Pasteris, P., Vallero, S., Rabellino, S.: The Open Computing Cluster for Advanced data Manipulation (OCCAM). In: The 22nd International Conference on Computing in High Energy and Nuclear Physics (CHEP). San Francisco, USA (2016)
2. Audrito, G., Beal, J., Damiani, F., Pianini, D., Viroli, M.: The share operator for field-based coordination. In: Coordination Models and Languages (COORDINATION). Lecture Notes in Computer Science, vol. 11533, pp. 54–71. Springer (2019). https://doi.org/10.1007/978-3-030-22397-7_4
3. Audrito, G., Beal, J., Damiani, F., Viroli, M.: Space-time universality of field calculus. In: Coordination Models and Languages (COORDINATION). Lecture Notes in Computer Science, vol. 10852, pp. 1–20. Springer (2018). https://doi.org/10.1007/978-3-319-92408-3_1

4. Audrito, G., Bergamini, S.: Resilient blocks for summarising distributed data. In: 1st Workshop on Architectures, Languages and Paradigms for IoT (ALP4IoT). pp. 23–26 (2017). https://doi.org/10.4204/EPTCS.264.3
5. Audrito, G., Bergamini, S., Damiani, F., Viroli, M.: Effective collective summarisation of distributed data in mobile multi-agent systems. In: 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS). pp. 1618–1626. IFAAMAS (2019)
6. Audrito, G., Casadei, R., Damiani, F., Viroli, M.: Compositional blocks for optimal self-healing gradients. In: Self-Adaptive and Self-Organizing Systems (SASO). pp. 91–100. IEEE (2017). https://doi.org/10.1109/SASO.2017.18
7. Audrito, G., Damiani, F., Viroli, M.: Aggregate graph statistics. In: 1st Workshop on Architectures, Languages and Paradigms for IoT (ALP4IoT). pp. 18–22 (2017). https://doi.org/10.4204/EPTCS.264.2
8. Audrito, G., Damiani, F., Viroli, M.: Optimally-self-healing distributed gradient structures through bounded information speed. In: Coordination Models and Languages (COORDINATION). Lecture Notes in Computer Science, vol. 10319, pp. 59–77. Springer (2017). https://doi.org/10.1007/978-3-319-59746-1_4
9. Audrito, G., Damiani, F., Viroli, M.: Optimal single-path information propagation in gradient-based algorithms. Sci. Comput. Program. **166**, 146–166 (2018). https://doi.org/10.1016/j.scico.2018.06.002
10. Audrito, G., Damiani, F., Viroli, M., Bini, E.: Distributed real-time shortest-paths computations with the field calculus. In: IEEE Real-Time Systems Symposium (RTSS). pp. 23–34. IEEE Computer Society (2018). https://doi.org/10.1109/RTSS.2018.00013
11. Audrito, G., Viroli, M., Damiani, F., Pianini, D., Beal, J.: A higher-order calculus of computational fields. ACM Transactions on Computational Logic **20**(1), 5:1–5:55 (2019). https://doi.org/10.1145/3285956
12. Beal, J.: Flexible self-healing gradients. In: ACM Symposium on Applied Computing (SAC). pp. 1197–1201. SAC '09, ACM (2009). https://doi.org/10.1145/1529282.1529550
13. Beal, J., Michel, O., Schultz, U.P.: Spatial computing: Distributed systems that take advantage of our geometric world. ACM Transactions on Autonomous and Adaptive Systems **6**(2), 11:1–11:3 (2011). https://doi.org/10.1145/1968513.1968514
14. Beal, J., Pianini, D., Viroli, M.: Aggregate programming for the internet of things. IEEE Computer **48**(9), 22–30 (2015). https://doi.org/10.1109/MC.2015.261
15. Beal, J., Usbeck, K., Loyall, J., Rowe, M., Metzler, J.: Adaptive opportunistic airborne sensor sharing. ACM Transactions on Autonomous and Adaptive Systems **13**(1), 6:1–6:29 (2018). https://doi.org/10.1145/3179994
16. Bicocchi, N., Mamei, M., Zambonelli, F.: Self-organizing virtual macro sensors. ACM Transactions on Autonomous and Adaptive Systems **7**(1), 2:1–2:28 (2012). https://doi.org/10.1145/2168260.2168262
17. Casadei, R., Pianini, D., Viroli, M., Natali, A.: Self-organising coordination regions: A pattern for edge computing. In: Coordination Models and Languages (COORDINATION). Lecture Notes in Computer Science, vol. 11533, pp. 182–199. Springer (2019). https://doi.org/10.1007/978-3-030-22397-7_11
18. Coutaz, J., Crowley, J.L., Dobson, S., Garlan, D.: Context is key. Communications of the ACM **48**(3), 49–53 (2005). https://doi.org/10.1145/1047671.1047703
19. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008). https://doi.org/10.1145/1327452.1327492

20. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Transactions on Computer Systems **23**(3), 219–252 (2005). https://doi.org/10.1145/1082469.1082470
21. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978). https://doi.org/10.1145/359545.359563
22. Liu, Q., Pruteanu, A., Dulman, S.: Gradient-based distance estimation for spatial computers. Comput. J. **56**(12), 1469–1499 (2013). https://doi.org/10.1093/comjnl/bxt124
23. Lluch-Lafuente, A., Loreti, M., Montanari, U.: Asynchronous distributed execution of fixpoint-based computational fields. Logical Methods in Computer Science **13**(1) (2017). https://doi.org/10.23638/LMCS-13(1:13)2017
24. Mamei, M., Zambonelli, F., Leonardi, L.: Co-fields: A physically inspired approach to motion coordination. IEEE Pervasive Computing **3**(2), 52–61 (2004). https://doi.org/10.1109/MPRV.2004.1316820
25. Moustafa, H., Zhang, Y.: Vehicular Networks: Techniques, Standards, and Applications. Auerbach Publications, Boston, MA, USA, 1st edn. (2009)
26. Nath, S., Gibbons, P.B., Seshan, S., Anderson, Z.R.: Synopsis diffusion for robust aggregation in sensor networks. TOSN **4**(2), 7:1–7:40 (2008). https://doi.org/10.1145/1340771.1340773
27. Pianini, D., Montagna, S., Viroli, M.: Chemical-oriented simulation of computational systems with ALCHEMIST. J. Simulation **7**(3), 202–215 (2013). https://doi.org/10.1057/jos.2012.27
28. Pianini, D., Viroli, M., Beal, J.: Protelis: Practical aggregate programming. In: ACM Symposium on Applied Computing (SAC). pp. 1846–1853 (2015). https://doi.org/10.1145/2695664.2695913
29. Talele, A.K., Patil, S.G., Chopade, N.B.: A survey on data routing and aggregation techniques for wireless sensor networks. In: International Conference on Pervasive Computing (ICPC). pp. 1–5. IEEE (2015)
30. Viroli, M., Audrito, G., Beal, J., Damiani, F., Pianini, D.: Engineering resilient collective adaptive systems by self-stabilisation. ACM Transactions on Modeling and Computer Simulation **28**(2), 16:1–16:28 (2018). https://doi.org/10.1145/3177774
31. Viroli, M., Beal, J., Damiani, F., Audrito, G., Casadei, R., Pianini, D.: From distributed coordination to field calculus and aggregate computing. J. Log. Algebraic Methods Program. **109** (2019). https://doi.org/10.1016/j.jlamp.2019.100486
32. Viroli, M., Damiani, F.: A calculus of self-stabilising computational fields. In: Coordination Languages and Models, Lecture Notes in Computer Science, vol. 8459, pp. 163–178. Springer-Verlag (2014). https://doi.org/10.1007/978-3-662-43376-8_11
33. Viroli, M., Pianini, D., Ricci, A., Croatti, A.: Aggregate plans for multiagent systems. International Journal of Agent-Oriented Software Engineering **4**(5), 336–365 (2017). https://doi.org/10.1504/IJAOSE.2017.087638
34. Zhang, Y., Lin, X., Yuan, Y., Kitsuregawa, M., Zhou, X., Yu, J.X.: Duplicate-insensitive order statistics computation over data streams. IEEE Trans. Knowl. Data Eng. **22**(4), 493–507 (2010). https://doi.org/10.1109/TKDE.2009.68
35. Zhou, G., He, T., Krishnamurthy, S., Stankovic, J.A.: Impact of radio irregularity on wireless sensor networks. In: 2nd International Conference on Mobile Systems, Applications, and Services. pp. 125–138. MobiSys '04, ACM, New York, NY, USA (2004). https://doi.org/10.1145/990064.990081