Optimal resilient distributed data collection in mobile edge environments

Optimal Resilient Distributed Data Collection in Mobile Edge Environments ⋆

Giorgio Audrito[a], Roberto Casadei[b], Ferruccio Damiani[a], Danilo Pianini[b], Mirko Viroli[b]

[a] *Università di Torino, Italy, {giorgio.audrito,ferruccio.damiani}@unito.it*
[b] *Alma Mater Studiorum—Università di Bologna, Italy, {roby.casadei,danilo.pianini,mirko.viroli}@unibo.it*

Abstract

A key goal of edge computing is to achieve "distributed sensing" out of data continuously generated from a multitude of interconnected physical devices. The traditional approach is to gather information into sparse collector devices by relying on hop-by-hop accumulation, but issues of reactivity and fragility naturally arise in scenarios with high mobility. We propose novel algorithms for dynamic data summarisation across space, supporting high reactivity and resilience by specific techniques maximising the speed at which information propagates towards collectors. Such algorithms support idempotent and arithmetic aggregation operators and, under reasonable network assumptions, are proved to achieve optimal reactivity. We provide evaluation via simulation: first in multiple scenarios showing improvement over the state of art, and then by a case study in edge data mining, which conveys the practical impact in higher-level distributed sensing patterns.

Keywords: Data Aggregation, Adaptive Algorithm, Aggregate Programming, Computational Field

1. Introduction

A key computational task in the Internet of Things (IoT) is extraction of valuable information from contextual data continuously generated by the multitude of smart sensors pervasively deployed in the physical environment [1], a problem made hard by requirements such as scalability, low-latency, adaptivity, and privacy. Altogether, they foster decentralised solutions at the edge of the network, complementing cloud-based solutions [2]: namely, as anticipated in contexts like Wireless Sensor Networks (WSNs) and spatial computing [3], forms of *distributed sensing* [4] are carried out through continuous cooperation and interaction among nearby devices, defining so-called "virtual macro sensors". In this view, devices are required to emit, process, and route information according to local knowledge, by collectively working as a *self-organising* system [5], i.e., a system sustaining its goal (order, structures, behaviour) – withstanding changes and environmental perturbations – by the continuous activity and local interaction of its internal components.

A fundamental pattern in such "cooperative" distributed sensing is *data summarisation*, by which the information dynamically sensed or produced by devices situated in a spatial region is aggregated into new

⋆This work has been supported by the MIUR PRIN 2017 Project N. 2017KRC7KT "Fluidware".

information, to be updated continuosly and automatically. This is at the basis of several higher-level tasks including counting items, measuring space, averaging environmental values, assessing the minimum or maximum levels of a physical quantity, tracking history, aggregate logging, and so on [6]. Notice that since sensing is distributed, it is generally not possible to compute "instantaneous" exact summaries: so, emphasis moves to so-called *self-stabilising* computations [7], which ensure to eventually produce correct results when inputs and perturbations stabilise, and to quickly produce "reasonable" *approximations* (where what is reasonable depends on the particular application).

Data summarisation can be implemented by algorithms of *distributed collection*, which instruct devices to progressively combine from (and propagate information to) neighbour devices, along the path (*single-path collection*) or paths (*multi-path collection*) towards the nearest *collector* device (also known as *sink*). They can hence be seen as a cooperative behaviour which takes as input: *(i)* the devices that must receive the collection result; *(ii)* the values produced by devices (sensors) that must be summarised; and *(iii)* the aggregation operator that must be used to combine the individual contributions with the partial collection result. At the implementation level, this algorithm typically requires a potential field [7, 8] used to give directions from sensors to the collectors, and progressively move information chunks toward it, and combine them (with an associative and commutative aggregation operator).

Existing solutions to the problem, however, assume devices are stationary or move very slowly, and hence are fragile to network dynamics [7], or deal with mobility using heuristics that might be partly effective [9], limiting their applicability in practice. To support effectiveness in mobile edge computing applications [2] (swarm robotics, crowd monitoring and control, vehicular networks), we propose novel algorithms specifically designed to address high variability, there including high speed mobility of devices and data loss. We focus on *idempotent* aggregation (e.g., minimum value computation, or union of sets) and *arithmetic* aggregation (e.g., sum computation), and achieve optimal reactivity to changes through the maximisation of *information speed* by choosing the "best" path selection strategies (encompassing both single-path and multi-path solutions). As such, we shall design our algorithms to be optimal.

In summary, we provide the following contributions (see also Table 1): *(1)* design C_{blist} algorithms for idempotent and arithmetic aggregations, which optimise under a statistical *bounded-loss* constraint; *(2)* evaluate them in synthetic scenarios varying relevant characteristic parameters (diameter, density, mobility); *(3)* demonstrate applicability on a realistic case study in edge data mining.

The rest of the manuscript is organised as follows: Section 2 provides background and state of the art in distributed data collection algorithms; Section 3 is the core of the contribution, describing the proposed algorithms; Section 4 provides a simulation-based evaluation of performance against state-of-the-art collection algorithms; Section 5 presents a realistic case study demonstrating applicability to edge data mining scenarios; and finally, Section 6 provides a wrap-up and points out opportunities for future work.

Assumptions	**State of the art** (Sec. 2)			**Proposed** (Sec. 3)	
	C_{sp}	C_{mp}	C_{wmp}	C_{list}	C_{blist}
Underlying model of computation (Section 2.2)	•	•	•	•	•
Based on a potential field (Section 2.3.1)	•	•	•	•	•
Basic assumptions (Section 3.1)				•	•
Assumption: sure connection (Section 3.1)				•	
Constraints (Section 3.2)					
Scalability	•	•	•	•	•
No data loss				•	
Bounded loss				•	•
Evaluation (Section 4)					
Best results (idempotent aggr., low mobility)				•	•
Best results (idempotent aggr., high mobility)			○	○	•
Best results (arithmetic aggr., static scenarios)	•			○	
Best results (arithmetic aggr., non-static scenarios)					•

Table 1: Comparison of the proposed collection algorithms (C_{list}, C_{blist}) w.r.t. state-of-the-art algorithms (C_{sp}, C_{mp}, C_{wmp}). Notation: • denotes a match; ○ denotes a close match.

2. Background and Related Work

We start defining background for the problem of collecting information available in spatially-distributed networks: Section 2.1 briefly recalls the mathematical fundamentals of distributed data collection; Section 2.2 presents the proximity-based interaction model we shall assume; and Section 2.3 exploits this model to illustrate state-of-the-art distributed data collection algorithms.

2.1. Mathematical Fundamentals of Distributed Data Collection

Data summarisation is a cornerstone routine of many distributed applications. As such, several algorithms realising it have been proposed for different scenarios like spatial computing, high-performance computing, and WSNs [3]. Despite differences in implementation details, every such algorithm is based on the same mathematical characterisation of the problem. Namely, distributed values are to be combined via an *aggregation operator* $\oplus$ for which the following properties hold: *(1) associativity*: $u \oplus (v \oplus w) = (u \oplus v) \oplus w$; *(2) commutativity*: $u \oplus v = v \oplus u$. If $\bigoplus$ enjoys such properties, then the aggregation of a multi-set $\mathcal{C}$, i.e. $\bigoplus \mathcal{C}$, is well-defined (i.e., the order in which the elements of $\mathcal{C}$ are considered for the aggregation is not significant). Common aggregation operators include the idempotent operators of minimum and maximum as well as the arithmetic operators of multiplication and addition. Also, in scenarios characterised by frequent changes in inputs and data transmission issues (e.g., delays, packet drops) like WSNs and mobile computing,

an additional property is often needed: *(3) continuity*: the effect of considering a percentage p of errors in the aggregation tends to zero as $p \to 0$. The aforementioned operators (for idempotent and arithmetic aggregation) also enjoy such a property, which however does not hold, e.g., for operators like modular sum (where even a single erroneous contribution could disrupt the whole aggregation result).

2.2. An Abstract Model of Computation for Proximity-based Interaction Networks

To raise the abstraction level when designing and reasoning about algorithms for distributed data collection, we shall adopt a so-called aggregate programming stance [10]. On the one hand, we see the distributed system as a network of (possibly mobile) devices with a dynamic topology, induced by the proximity (physical or logical) of devices, where each device can send/receive messages to/from devices in the neighbourhood. On the other hand, we shall specify an adaptive system for such networks in terms of a "global program", intended as single specification that each device repetitively execute, until the expected global result is reached. Namely, we assume an underlying model of computation where each device of the network asynchronously and periodically executes a task comprising the following steps: *(1)* collection of status of its sensors and messages received (to be used as input for the next step); *(2)* execution of a program (the same for all devices) yielding a value and messages to be sent; and *(3)* transmission of messages to neighbours. We say that a device *fires* to mean that it performs an execution of this task. A situation in which the task is not completed (e.g., the execution of the program in step 2 does not terminate) is *not* considered a firing.

The time and location of a firing is called an *event*, i.e., each event ϵ uniquely identifies the time instant $t(\epsilon)$, position in space $l(\epsilon)$, and the device $\delta(\epsilon)$ where a firing starts. For a set E of events, we write $\delta(E)$ as short for $\{\delta(\epsilon) \mid \epsilon \in E\}$. Events are related by message-passing, as described by the following definition.

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

In a real distributed system with asynchronous scheduling, a device could fire at a slower rate than another, thus receiving multiple messages from it during a single round: Definition 1 states that only on the last received may be considered. Conversely, no message from a "slow" device could reach a "fast" target during a round: according to Definition 1, devices may retain messages from neighbours across fires, increasing the computation stability (details on if and when messages are kept or discarded are system design choices).

In the remainder of this paper, we assume availability of the following quantities during each event ϵ on a device $\delta(\epsilon)$: *(1)* The radius R within which communication succeeds.[1] *(2)* The time difference (lag) with respect to each supplier event ϵ': $\mathrm{lag}(\epsilon', \epsilon) = t(\epsilon) - t(\epsilon')$.[2] *(3)* The measured distance with respect to

[1] In reality, the communication range of a node is very irregular. As suggested by Zhou et al. [11], such an irregular radius can be bounded below, justifying the usage of a fixed quantity.

[2] This quantity can be calculated with reasonable precision even without a global clock [12].

each supplier event ϵ': $\mathrm{dist}(\epsilon', \epsilon)$, possibly affected by errors. The latter quantity can be computed in three main different ways: *(1)* in GPS-based systems, $\mathrm{dist}(\epsilon', \epsilon)$ is (an approximation of) the distance between the positions $l(\epsilon')$ (referred to time $t(\epsilon')$) and $l(\epsilon)$ (referred to time $t(\epsilon)$); *(2)* if distance is sensed at message receipt, it is an approximation of the distance between devices $\delta(\epsilon')$ and $\delta(\epsilon)$ at time $t(\epsilon')$; *(3)* if distance can be sensed in every moment, then it is an approximation of the distance between devices at time $t(\epsilon)$.

The values manipulated by an aggregate program, which arise on each firing, can be modelled as functions of events: we will use $X(\epsilon)$ to denote such a distributed value X, and $X_{\epsilon'}(\epsilon)$ to denote a value depending on *supplying relationships* $\epsilon' \rightsquigarrow \epsilon$, that is, a quantity computed in ϵ with respect to a supplier event ϵ'.

2.3. State-of-the-art Distributed Data Collection Algorithms for Proximity-based Interaction Networks

In a proximity-based interaction network, given a commutative and associative operator, data collection algorithms asynchronously combine input values $x(\epsilon)$ from different devices into a single value in a selected *source* (or *collector*) device, managing data flow in order to avoid multiple aggregation of a same value. Such algorithms hence need to ensure that flows are *acyclic* and *directed* towards the source. This is achieved by relying on a *potential field* $P(\epsilon)$ approximating a measure of distance from the source: as information descends the potential field, cyclic dependencies are prevented and the source is eventually reached.

For each event ϵ, potential descent is enforced by splitting the set of supplier events $E_\epsilon = \{\epsilon' \mid \epsilon' \rightsquigarrow \epsilon\}$ into two disjoint sets $E_\epsilon^- = \{\epsilon' \in E_\epsilon \mid P(\epsilon') < P(\epsilon)\}$ and $E_\epsilon^+ = \{\epsilon' \in E_\epsilon \mid P(\epsilon') > P(\epsilon)\}$ of events with respectively lower and higher potential, while ensuring that values are received only from events in E_ϵ^+.

In the following, we survey algorithms for potential field computation (Section 2.3.1) and present the three main proximity-based interaction distributed collection algorithms: *single-path* (Section 2.3.2), *multi-path* (Section 2.3.3) and *weighted multi-path* (Section 2.3.4). All three algorithms scale to arbitrarily large systems, by requiring computational resources per device proportional to neighbourhood size. The second and third also address scenarios with ephemeral devices, while only the third addresses the issue of mobility.

2.3.1. Potential Field

A *potential field* $P(\epsilon)$ of distances from a *source* (or *sink*) node is a main input of every data aggregation algorithm, used to guide the collection flow towards that source. Accurately computing distances in a dynamic network is a demanding task, which can be tackled differently depending on the application scenario. In particular, it can be done in a decentralised, self-healing fashion using a so-called *gradient computation* [7], an algorithmic scheme (with various implementations) where devices compute and adaptively correct local distance estimates by combining neighbours' estimates with corresponding node-to-neighbour distances. These can be obtained by sensors (cf. Sec. 2.2) if available, or the basic *hop-count gradient* based on unit-distances can be used instead, possibly improved via statistical tools [13]. However, also when precise proximity estimates can be obtained, the correction of gradient estimates as sources change (input variability)

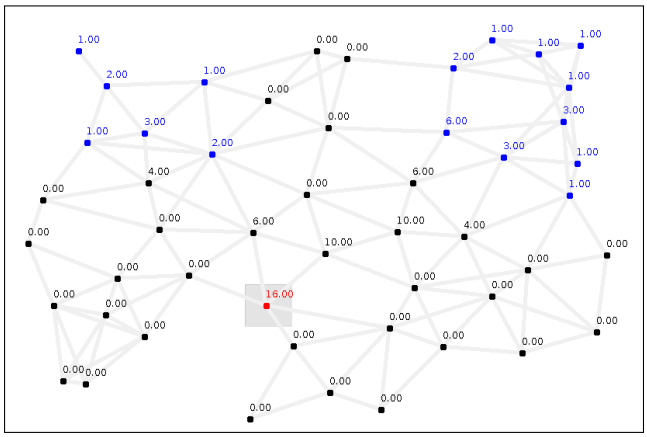


Figure 1: Result after convergence of a C_{sp} algorithm counting the number of blue nodes in the red *sink* node. Each node keeps a partial result of the process, achieved by aggregating data from the "single-path data flows" (from blue nodes to the red node) that pass through the node. Communication is bidirectional and the aggregation outcome grows towards the sink.

or nodes move (network variability) can be hindered by the *rising value problem*, whereby the system does not promptly react to changes causing distances to rise. To address this and other issues, various solutions have been suggested in literature: recent reviews [14, 15] point out three state-of-the-art solutions.

The FLEX algorithm [7] aims at maximising stability of distance estimates while bounding the error; and handles the rising value problem via metric distortion. The BIS algorithm [15] leverages time information to enforce a minimum information speed, hence achieving optimal single-path reactivity to input changes, including those causing values to rise. The ULT algorithm [14] builds on BIS by adding a detector for obsolete values operating at multi-path speed (i.e., faster), and by dealing with estimate stability via dampers and filters. Since it integrates many techniques, it also depends on many parameters which need to be tuned: depending on those, ULT behaviour can range from being closer to BIS to being closer to FLEX.

2.3.2. Single-path Collection

In the single-path algorithm C_{sp}, at each event ϵ, the firing device sends the partial aggregate $C_{\text{sp}}(\epsilon)$ just computed to a single device: the one that fired at the supplier event $m(\epsilon) = \epsilon'$ with minimum potential $P(\epsilon')$ among all supplier events in E_ϵ. In formulas, assuming that $x(\epsilon)$ is the value to be collected:

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

Equation 1 specifies that the partial aggregate in ϵ must be computed by combining the local input value $x(\epsilon)$ and the partial aggregates from supplier events ϵ' with higher potential for which $\delta(\epsilon)$ is the selected device $\delta(m(\epsilon'))$. This ensures that information flows through a forest in the network. A screenshot illustrating, for each device, the value computed by this algorithm after convergence is shown in Figure 1.

As information descends the potential with maximal flowing speed, single-path aggregation achieves optimal reactivity to input changes for networks with static topology [9]. However, in networks with ephemeral devices and high variability, the potential loss of the message from ϵ to $m(\epsilon)$ can cause the whole branch of the forest rooted at ϵ to be ignored by the algorithm, resulting into poor performances.

2.3.3. Multi-path Collection

The multi-path algorithm C_{mp} prescribes that, at each event ϵ, the firing device divides the partial aggregate $C_{\text{mp}}(\epsilon)$ equally among *every* device that fired at a supplier event ϵ' with lower potential. Thus, data flows through every path compatible with the given potential field. This is achieved by computing:

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

where $N(\epsilon) = |E_\epsilon^-|$ and $v \oslash n$ corresponds to the "division of v by n": it yields the element such that, when aggregated n times with itself, produces v. Operator $\oslash$ can only be defined for "divisible" data, which may be of two main kinds: *(1) idempotent operators*, such as those computing the minimum or maximum of a partially ordered set (for which $\oslash$ is the identity function); *(2) arithmetic operators*, such as point-wise multiplication and addition of vectors of real numbers (where $\oslash$ is division and root extraction, respectively). This requirement restricts the scope of multi-path aggregation, however, most operators recurring in practice can be expressed in terms of idempotent or arithmetic operations. For instance, idempotent operators combined with statistical tools can be used to emulate various kinds of aggregations including count of distinct values, addition, uniform sampling, frequent values selection, and order statistics [16].

As information flows through every path, devices are unlikely to be excluded from the aggregation, reducing data loss in presence of ephemeral devices. However, this also impairs reactivity to input changes: even assuming a static topology, data flows on every path including the *longest*, delaying the reaction until all paths have been considered (especially for idempotent operations), scoring a reaction speed inversely proportional to network density. In presence of mobile devices, the issue is made worse by the formation of *information loops*, occurring when devices of similar potential invert (by moving) their relative ordering in subsequent firings, causing information from a device to return to itself. This phenomenon further reduces the reactivity of the algorithm, and induces exponential overestimations in arithmetic data collection.

2.3.4. Weighted Multi-path Collection.

In the weighted multi-path algorithm C_{wmp} [9], at each event ϵ, the firing device divides *unevenly* the partial aggregate $C_{\text{wmp}}(\epsilon)$ among devices that fired at a supplier event ϵ' with lower potential, by assigning them *weights* designed to penalise devices that are likely to lose their "receiving" status. This can occur either: *(1)* if the "recipient" is far from the "sending" device, so that their connection can soon break; or *(2)* if the potential values of the "recipient" and "sending" devices are too close, so that their role could

soon be exchanged, allowing for an "information loop" between them. Both cases are dealt with a weight function $w_{\epsilon'}(\epsilon) = d(\epsilon', \epsilon) \cdot p(\epsilon', \epsilon)$, determining how much data from ϵ should be transmitted to a neighbour $\delta(\epsilon')$. The function is built on two components, $d(\epsilon', \epsilon) = R - \mathrm{dist}(\epsilon', \epsilon)$ and $p(\epsilon', \epsilon) = |P(\epsilon) - P(\epsilon')|$, where R is the communication radius and $\mathrm{dist}(\epsilon', \epsilon)$ is a measure of distance between ϵ' and ϵ. Such weights have to be normalised by factor $N(\epsilon) = \sum_{\epsilon' \in E_\epsilon^-} w_{\epsilon'}(\epsilon)$, obtaining normalised weights $w_{\epsilon'}(\epsilon)/N(\epsilon)$. Any device can then compute its partial aggregate as in C_{mp} (cf. Equation (2)) with the addition of weights:

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

where binary operator $\otimes$ above is such that $v \otimes k$ "extracts" from a local value v a percentage k.[3] For instance, if $\oplus$ is addition then $\otimes$ is multiplication, while if $\oplus$ is idempotent then $\otimes$ is a threshold function determining which links should be exploited or not for sending data.

It has been shown that C_{wmp} largely outperforms both C_{sp} and C_{mp} [9]. However, it leverages heuristics and hence correctness guarantees cannot be provided. Furthermore, it suffers in scenarios of arithmetic data collection characterised by high mobility, where the system undergoes exponentially erroneous behaviour [9].

3. Collection by Information Speed Thresholds

This section introduces the *Lossless Information Speed Thresholds* (C_{list}) and *Bounded-Loss Information Speed Thresholds* (C_{blist}) collection algorithms. These algorithms work by maximising the information speed on the basis of the assumptions described in Section 2.2 and the further network model assumptions described in Section 3.1, while satisfying the constraints presented in Section 3.2.

3.1. Network Model Assumptions

We consider a potential field $P(\epsilon)$ for any event ϵ as input (cf. Sec. 2), and use notation ϵ_{next} for the event that follows ϵ on the same device: $\epsilon \rightsquigarrow \epsilon_{\mathrm{next}}$ and $\delta(\epsilon) = \delta(\epsilon_{\mathrm{next}})$. In order to compute C_{list} and C_{blist}, we leverage some basic forecasting of values in subsequent events ϵ_{next}, enabled by the following assumptions.

Scheduled time. For any event ϵ, we require a known upper bound $t^u(\epsilon)$ to $t(\epsilon_{\mathrm{next}})$. This requirement can be satisfied easily and precisely, since the scheduling of firings is generally controlled by a large degree.

Potential field dynamics. For any event ϵ, we require a known upper and lower bounds $P^u(\epsilon)$, $P^l(\epsilon)$ to $P(\epsilon_{\mathrm{next}})$. For instance, if V is an upper bound on the movement speed of devices, we can define $P^u(\epsilon) = P(\epsilon) + V \cdot (t^u(\epsilon) - t(\epsilon))$. Such an upper bound may be adjusted by accounting for errors in potential field computation, and could be refined if the movement direction can be guessed. For probability computations, we assume $P(\epsilon_{\mathrm{next}})$ to vary uniformly at random between its upper and lower bounds.[4]

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

[4] Though this model is not very realistic, it is "pessimistic": values are usually more concentrated towards the middle.

Communication success. For each event ϵ and neighbour ϵ', there is a conservative (and reasonably accurate) estimate $\text{connectionSuccess}_{\epsilon'}(\epsilon)$ of the probability with which ϵ is able to successfully send a message to the next event ϵ'_{next} on $\delta(\epsilon')$.

Sure connection. Additionally, C_{list} requires that some neighbour can *surely* receive messages. We define $\text{surelyConnected}_{\epsilon'}(\epsilon)$ as the Boolean value of whether $\text{connectionSuccess}_{\epsilon'}(\epsilon)$ is equal to 1, and assume that for each event ϵ there is a neighbour ϵ' satisfying $\text{surelyConnected}_{\epsilon'}(\epsilon)$. Notice that surelyConnected could be obtained from an upper bound on distance $\text{dist}(\epsilon', \epsilon)$, an upper bound on speed V, and a lower bound on communication radius R:

$$\text{surelyConnected}_{\epsilon'}(\epsilon) \Leftrightarrow \text{maxDistNow}(\epsilon', \epsilon) \leq R, \qquad \text{maxDistNow}(\epsilon', \epsilon) = \text{dist}(\epsilon', \epsilon) + kV\,\text{lag}(\epsilon', \epsilon)$$

where k is set to 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 $t(\epsilon')$ (cf. Section 2.2). Estimating connectionSuccess requires an additional knowledge of the network model. For instance, if we assume: *(1)* a probability $F(d)$ of communication success for devices at a distance d, with integral $F^i(d) = \int F(d)\partial d$; *(2)* that the relative distance of devices varies between its minimum and maximum bounds uniformly at random; then $\text{connectionSuccess}_{\epsilon'}(\epsilon)$ could be estimated as:

$$\text{connectionSuccess}_{\epsilon'}(\epsilon) \backsimeq \frac{F^i(\text{maxDistNow}(\epsilon', \epsilon)) - F^i(\text{minDistNow}(\epsilon', \epsilon))}{\text{maxDistNow}(\epsilon', \epsilon) - \text{minDistNow}(\epsilon', \epsilon)}.$$

As an example, consider a crowd tracking system estimating count and barycentre of a crowd in a public event in a decentralised fashion, possibly using low-power wearable devices (e.g., bracelets) provided by organisers. For such a system, it would be reasonable to meet the above requirements by investigating the networking system properties prior to deployment, as radios and protocols are known in advance.

3.2. Algorithmic Constraints

In addition to the assumptions above, we restrict our focus to *scalable* and *lossless* or *bounded-loss* collection algorithms. A distributed algorithm is said to be *scalable* if, in any event, partial results require $O(1)$ message size and $O(N)$ time and space complexity to be computed, where $N = |E_\epsilon|$ denotes the size of the neighbourhood of ϵ, namely, the number of its supplier events.

A data collection algorithm is said to be *lossless* if it guarantees that the input value $x(\epsilon)$ in any event contributes to the algorithm outcome $C(\epsilon')$ for at least one event ϵ' in the *collector* node where $P(\epsilon') = 0$. This will be shown for C_{list} by inductively ensuring that the partial result of an event is received and considered by at least one neighbour event. In scenarios with high mobility or unreliability, this condition may be impossible to meet. Thus, we also consider (for C_{blist}) the following more relaxed constraint.

A collection algorithm has *bounded-loss* if for every event ϵ the "relative loss" random variable X_ϵ has mean 1 and variance below a given threshold; where X_ϵ corresponds to the relative fraction of the partial result in ϵ that is actually received by any neighbour. For arithmetic aggregations, this relative fraction

for a reference event can be straightforwardly computed as $\frac{x_1+\ldots+x_n}{x}$, where x is the partial result in the reference event ϵ, and $x_1, \ldots, x_n$ are the values received and used by neighbours of ϵ to compute their partial results. For idempotent aggregations, there is no notion of "fraction" between values; instead, values are either received and used or not: we say that the fraction is 1 in the affirmative case and 0 in the negative case. In this case, the relative loss random variable X_ϵ degenerates to a Bernoulli distribution; and bounding its variance is equivalent to bounding the probability for X_ϵ to be zero. Notice that in several real world scenarios of aggregation, such as the crowd density and barycentre estimation, there is typically a need for a reasonably reliable estimate rather than a exact, lossless value. It is key, though, to have the ability to bound the error, so that the estimate can be correctly interpreted by domain experts.

3.3. Lossless Idempotent Aggregation

For idempotent aggregation, double counting of the same contribution has no negative impact on the overall collection result, so emphasis can be put on avoiding loss of data by leveraging multi-path strategies (i.e., sending data to multiple neighbours). Since the basic multi-path algorithm is unable to quickly self-adjust [9], we need to tune it in order to enforce the maximum *information speed* v (as potential descended by messages per time unit), exploiting the network model assumptions defined in Section 3.1. Indeed, if the algorithm does not consider the longer paths from any event ϵ to the collector (i.e., it does not consider obsolete information), then reactivity to input is maximised for quick revision of the results.

Note that a *scalable* algorithm cannot be based on the information speed v of entire paths, since it would be impossible to select them in intermediate events according to this criterion. For any set of values k reaching an event ϵ through potential ΔP_k in a timespan Δt_k, a constant-sized subset of those values must be selected without knowledge of the supplemental timespan Δt required to reach the collector, and hence of the speed associated with the entire paths of the candidate values. Therefore, we implicitly choose paths by setting a speed constraint to *each* edge. The *threshold speed* $\theta(\epsilon)$ of an event ϵ is the speed that must be reached by a message $\epsilon \rightsquigarrow \epsilon'$ in order to be considered:

$$v(\epsilon, \epsilon') = \frac{P(\epsilon) - P(\epsilon')}{t(\epsilon') - t(\epsilon)} < \theta(\epsilon) \tag{4}$$

In other words, if information transmitted by ϵ to ϵ' descends the potential $P(\cdot)$ at a speed lower than the threshold determined in $\theta(\epsilon)$, then it is discarded. Since a global threshold may not take into account the variety of situations arising in the whole network, potentially inducing data loss for substantial portions of it, we exploit local thresholds computed in each event. Additionally, such thresholds are computed to be maximal (in order to discard as many paths as possible) while ensuring that the message will be considered by at least one device in the neighbourhood; hence allowing the algorithm to be *lossless*.

For effective and efficient choice of such thresholds, we exploit the model assumptions of Section 3.1. To prevent data loss, for any event ϵ we must ensure at least one neighbour event $\epsilon' \rightsquigarrow \epsilon$ for which

surelyConnected$_{\epsilon'}(\epsilon)$ holds does consider the message. Through the upper bounds $P^u(\cdot)$ on potential and $t^u(\cdot)$ on the scheduling time, we can set a lower bound on the information speed from ϵ to ϵ'_{next}:

$$v(\epsilon, \epsilon'_{\text{next}}) = \frac{P(\epsilon) - P(\epsilon'_{\text{next}})}{t(\epsilon'_{\text{next}}) - t(\epsilon)} \geq \frac{P(\epsilon) - P^u(\epsilon')}{t^u(\epsilon') - t(\epsilon)} = v^{\text{wst}}_{\epsilon'}(\epsilon) \tag{5}$$

Therefore, the maximum threshold granting zero loss of data is defined as follows:[5]

$$\theta(\epsilon) = \max\left\{v^{\text{wst}}_{\epsilon'}(\epsilon) : \ \text{surelyConnected}_{\epsilon'}(\epsilon) = \top\right\} \tag{6}$$

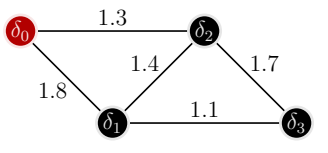
So, partial accumulation results can be computed in any event ϵ by repeated application of the update rule:

$$C_{\text{list}}(\epsilon) = x(\epsilon) \oplus \bigoplus_{\epsilon' \in E_\epsilon} \left\{C_{\text{list}}(\epsilon') : \ v(\epsilon', \epsilon) = \frac{P(\epsilon') - P(\epsilon)}{t(\epsilon) - t(\epsilon')} \geq \theta(\epsilon')\right\} \tag{7}$$

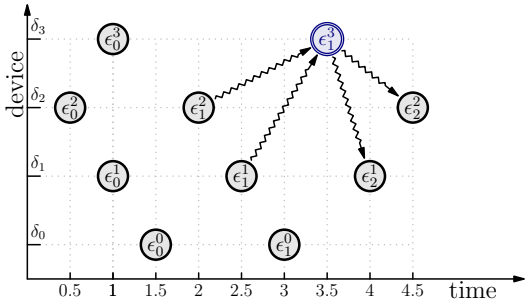
where the locally available contribution $x(\epsilon)$ is aggregated with the partial collection results provided only by those supplier events ϵ' for which the speed $v(\epsilon', \epsilon)$ of information flowing to ϵ exceeds the threshold determined by previous events $\theta(\epsilon')$. The C_{list} algorithm is thus altogether defined by Equations (5) to (7). Even though the thresholds determined at any event maximise the expected future information speed, and the neighbour that theoretically provides the highest speed is accordingly selected, the C_{list} algorithm is not single-path: indeed, since a message $\epsilon \rightsquigarrow \epsilon'_{\text{next}}$ can go faster than the estimated lower bound $v^{\text{wst}}_{\epsilon'}(\epsilon)$ (cf. Equation (5)), it can still exceed the threshold $\theta(\epsilon)$ (cf. Equation (6)) designed for another target. As per the above explanation, we can finally state the following property.

Property 1 (C_{list} is locally optimal among lossless collection algorithms). *Let threshold $\theta(\epsilon)$ be such that, using information available at event ϵ, it is possible to guarantee a lowest speed of information coming out from ϵ of at least $\theta(\epsilon)$ without risk of losing data. Then, the lowest speed of information coming out from ϵ for C_{list} is greater or equal to $\theta(\epsilon)$.*

Example 1. *Consider the following network (left), where each device δ_i performs events ϵ_i^j according to the given time schedule (right):*



	δ_0	δ_1	δ_2	δ_3
$P(\cdot)$	0.0	1.8	1.3	2.9



[5] If surelyConnected$_{\epsilon'}(\epsilon)$ does not hold for any neighbour, it is not possible to ensure zero loss of data and we set the threshold to $-\infty$, hence settling for a gossip-like behaviour.

For simplicity, we assume that the potential in each event equals the exact distance in the graph from the source, and its lower and upper bounds are ± 0.5 *away from it. Similarly, we assume that* $t(\cdot)$ *and* $t^u(\cdot)$ *correspond to the exact timing of their current (resp. next) event, and that all devices are surely connected. Consider event* ϵ_1^3 *with suppliers* ϵ_1^1, ϵ_1^2. *We have that:* $v^{wst}_{\epsilon_1^1}(\epsilon_1^3) = {}^{(2.9-2.3)}/_{(4.0-3.5)} = 1.2$ *and* $v^{wst}_{\epsilon_1^2}(\epsilon_1^3) = {}^{(2.9-1.8)}/_{(4.5-3.5)} = 1.1$. *Thus,* $\theta(\epsilon_1^1) = \max(1.2, 1.1) = 1.2$. *In event* ϵ_2^1, *the data from* ϵ_1^3 *will be considered since it will be having speed* ${}^{(2.9-1.8)}/_{(4-3.5)} = 2.2 > 1.2$. *In event* ϵ_2^2, *the data from* ϵ_1^3 *will also be considered since it will be having speed* ${}^{(2.9-1.3)}/_{(4.5-3.5)} = 1.6 > 1.2$.

3.4. Bounded-Loss Idempotent Aggregation

In settings for which the *lossless* constraint is not satisfiable, we consider the weaker *bounded-loss* constraint (cf. Section 3.2). In the idempotent case, this requires the probability of total loss for each event to be below a fixed threshold p_{fail}. As in the lossless case, we resort to a multi-path strategy, pruning the slowest communication paths while respecting the constraint. Thus, we need to calculate a *threshold speed* $\theta(\epsilon)$ in event ϵ, regulating message discard as in Equation (4). Given such a threshold speed θ as a parameter, by Equation (4), the message from ϵ for $\delta(\epsilon')$ is not discarded provided that the potential $P(\epsilon'_{\text{next}})$ satisfies:

$$\theta \leq \frac{P(\epsilon) - P(\epsilon'_{\text{next}})}{t(\epsilon'_{\text{next}}) - t(\epsilon)} \quad \Leftrightarrow \quad P(\epsilon'_{\text{next}}) \leq P(\epsilon) - \theta(t(\epsilon'_{\text{next}}) - t(\epsilon)).$$

Since $t(\epsilon'_{\text{next}}) \leq t^u(\epsilon')$, a sufficient condition is $P(\epsilon'_{\text{next}}) \leq P(\epsilon) - \theta(t^u(\epsilon') - t(\epsilon))$. Assuming that $P(\epsilon'_{\text{next}})$ varies uniformly at random between its bounds $P^l(\epsilon')$ and $P^u(\epsilon')$ (cf. Section 3.1), we obtain that a conservative estimate of the probability for the message not being discarded by the algorithm is:

$$\text{survivalP}(\epsilon, \epsilon', \theta) = \frac{P(\epsilon) - \theta \cdot (t^u(\epsilon') - t(\epsilon)) - P^l(\epsilon')}{P^u(\epsilon') - P^l(\epsilon')}.$$

Combining this with the probability $\text{connectionSuccess}(\epsilon, \epsilon')$ for the message not physically reaching the neighbour, we obtain that the overall probability of failure for the communication from ϵ to $\delta(\epsilon')$ is:

$$\text{failingP}(\epsilon, \epsilon', \theta) = 1 - \text{connectionSuccess}(\epsilon, \epsilon') \, \text{survivalP}(\epsilon, \epsilon', \theta) \tag{8}$$

and the probability of all communication failing together is $\text{failingP}(\epsilon, \theta) = \prod_{\epsilon' \rightsquigarrow \epsilon} \text{failingP}(\epsilon, \epsilon', \theta)$. In order to optimise the recovery speed of the algorithm under the bounded-loss constraint, we thus pose $\theta(\epsilon) = \max\{\theta \in [0, \infty] : \ \text{failingP}(\epsilon, \theta) \leq p_{\text{fail}}\}$. Notice that $\text{failingP}(\epsilon, \theta)$ is an increasing function of θ: when the threshold increases, every $\text{survivalP}(\epsilon, \epsilon', \theta)$ decreases hence $\text{failingP}(\epsilon, \epsilon', \theta)$ increases. Then, $\theta(\epsilon)$ can be effectively found by binary search with arbitrary precision. Besides the different criterion for computing $\theta(\epsilon)$, the C_{blist} algorithm is then identical to C_{list} as described in Equation (7):

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

Example 2. *Consider the network and events of Example 1, with the same potentials and upper and lower bounds. Assume that devices within a distance of 1.5 are surely connected, while more distant devices are 50% connected. Consider event* ϵ_1^3 *with suppliers* ϵ_1^1, ϵ_1^2. *Given* θ, *we have that:* $\text{survivalP}(\epsilon_1^3, \epsilon_1^1, \theta) = \frac{2.9-\theta(4.0-3.5)-1.3}{2.3-1.3} = 1.6 - \theta/2$ *and* $\text{survivalP}(\epsilon_1^3, \epsilon_1^2, \theta) = \frac{2.9-\theta(4.5-3.5)-0.8}{1.8-0.8} = 2.1 - \theta$. *Combining this with connection probabilities, we get that* $\text{failingP}(\epsilon_1^3, \theta) = (1 - 1.0(1.6 - \theta/2))(1 - 0.5(2.1 - \theta)) = (\theta/2 - 0.6)(\theta/2 - 0.05) = 0.25\theta^2 - 0.325\theta + 0.03$. *Assuming we want* $p_{fail} = 2.36\%$, *the highest* θ *we can choose is* $\theta(\epsilon_1^3) = 1.28$. *Both events* ϵ_2^1, ϵ_2^2 *will then consider data from* ϵ_1^3, *as the speed attained for them is* $2.2, 1.6 > 1.28$.

3.5. Lossless Arithmetic Aggregation

For arithmetic collection, we must prevent counting a same contribution more than once (*double counting*), to avoid an exponential accumulation of errors. Therefore, we amend C_{list} to work in a single-path fashion. Consider information speed conditions and maximum thresholds, eqs. (4) to (6). The single neighbour $m(\epsilon)$ that is chosen as recipient of the local partial collection is the one ensuring maximum $v^{\text{wst}}_{m(\epsilon)}(\epsilon)$:

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

If no neighbour is surely connected, data loss cannot be avoided and we choose $m(\epsilon)$ minimising the chance of losing data. Finally, the accumulation of partial aggregation results is like for C_{sp} (cf. Equation (1)):

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

That is, the locally provided value $x(\epsilon)$ is combined with partial accumulation results $C_{\text{list}}(\epsilon')$ of parent events ϵ', i.e., from those that chose the current device $\delta(\epsilon)$ as single-path recipient, $\delta(m(\epsilon'))$.

Example 3. *Consider the scenario described in Example 1. In event* ϵ_1^3, *the neighbour ensuring maximum* $v^{wst} = 1.2$ *is* $m(\epsilon_1^3) = \epsilon_1^1$. *Thus, event* ϵ_2^1 *will consider the data from* ϵ_1^3, *while* ϵ_2^2 *will not.*

3.6. Bounded-Loss Arithmetic Aggregation

If we tolerate some statistical amount of error, a better multi-path strategy can be devised for arithmetic aggregation. Suppose that a value x needs to be sent to neighbours $i = 1 \ldots n$, each of them with a probability p_i of receiving (and not discarding) a message. Let w_i for $i = 1 \ldots n$ be any positive weights assigned to those neighbours (not necessarily normalised), regulating how much of x needs to be transmitted to each of them. Let B_p be denoting a Bernoulli random variable with probability p of being 1, and $1 - p$ of being 0. Then, we can express the random variable X of the amount received by neighbours as $X = \sum_{i=1}^{n} x w_i B_{p_i}$. For the *bounded-loss* constraint (cf. Section 3.2), we need to find an assignment to w_i granting us that the mean of values received is indeed x. The average amount received by neighbours is $E(X) = xS_1$, where $S_1 = \sum_{i=1}^{n} w_i p_i$. It follows that in order to get an average value received of x, weights (hence X) need to normalised by dividing them by S_1, obtaining $w'_i = {}^{w_i}/_{S_1}$, $X' = {}^{X}/_{S_1}$.

For the *bounded-loss* constraint, we also need to bound the variance of the relative loss random variable, which in this case corresponds to $Y = {}^{X'}\!/_{x} = \frac{\sum_{i=1}^{n} x w_i' B_{p_i}}{x} = \sum_{i=1}^{n} w_i' B_{p_i} = \sum_{i=1}^{n} \frac{w_i}{S_1} B_{p_i}$. Then, its variance is $V(Y) = \sum_{i=1}^{n} \left(\frac{w_i}{S_1}\right)^2 V(B_{p_i}) = \sum_{i=1}^{n} \frac{w_i^2}{S_1^2} p_i(1-p_i) = \frac{S_2}{S_1^2}$, where $S_2 = \sum_{i=1}^{n} w_i^2 p_i(1-p_i)$. Since we need to bound $V(Y)$, we first need to compute an assignment of weights w_i minimising it. This can be obtained by posing the partial derivatives ${}^{\partial V(Y)}\!/_{\partial w_i}$ to be simultaneously zero:

$$\frac{\partial V(Y)}{\partial w_i} = \frac{\frac{\partial S_2}{\partial w_i} \cdot S_1^2 - S_2 \cdot 2S_1 \frac{\partial S_1}{\partial w_i}}{S_1^4} = \frac{S_1 \frac{\partial S_2}{\partial w_i} - 2S_2 \frac{\partial S_1}{\partial w_i}}{S_1^3} = \frac{S_1 \frac{\partial}{\partial w_i} \sum_{j=1}^{n} (w_j^2 p_j(1-p_j)) - 2S_2 \frac{\partial}{\partial w_i} \sum_{j=1}^{n} (w_j p_j)}{S_1^3}$$
$$= \frac{S_1 2 w_i p_i(1-p_i) - 2S_2 p_i}{S_1^3} = \frac{2p_i}{S_1^3}(S_1 w_i(1-p_i) - S_2) = 0 \Leftrightarrow S_1 w_i(1-p_i) = S_2 \Leftrightarrow w_i = \frac{S_2}{S_1(1-p_i)}$$

Although $\frac{S_2}{S_1}$ depends on $w_1, \ldots, w_n$, it does not depend on i: thus, weights are proportional to $\frac{1}{1-p_i}$ by the equation above. In fact, *any* assignment of weights proportional to $\frac{1}{1-p_i}$ satisfies the equation, since they are normalised to w_i' before being used in Y. This can be directly checked by posing $w_i = \frac{\alpha}{1-p_i}$ in the derivatives, obtaining ${}^{\partial V(Y)}\!/_{\partial w_i} = \frac{2p_i}{S_1^3}\left(S_1 \frac{\alpha}{1-p_i}(1-p_i) - S_2\right) = \frac{2p_i}{S_1^3} \sum_{j=1}^{n} (\alpha \frac{\alpha}{1-p_j} p_j - \frac{\alpha^2}{(1-p_j)^2} p_j(1-p_j)) = 0$. Since all derivatives are zero, it follows that this choice of w_i is a minimum, maximum or saddle point for $V(Y)$. Given that $V(Y)$ is a positive function of w_i, there has to exist a minimum which can either be the only point we found where derivatives vanish, or it may be a point at an edge of the domain. The points at the edge of the domain are those of the kind $w_j = 1$, $w_i = 0$ for $i \neq j$ (sending all data to neighbour j). The variance for one of these points is $V(Y) = {}^{S_2}\!/_{S_1^2} = \frac{\sum_{i=1}^{n} w_i^2 p_i(1-p_i)}{\left(\sum_{i=1}^{n} w_i p_i\right)^2} = \frac{p_j(1-p_j)}{p_j^2} = \frac{1}{p_j/(1-p_j)}$. Instead, the variance with $w_i = {}^{\alpha}\!/_{(1-p_i)}$ is $V(Y) = \frac{\sum_{i=1}^{n} (\alpha/(1-p_i))^2 p_i(1-p_i)}{\left(\sum_{i=1}^{n} \alpha p_i/(1-p_i)\right)^2} = \frac{\alpha^2 \sum_{i=1}^{n} p_i/(1-p_i)}{\alpha^2 \left(\sum_{i=1}^{n} p_i/(1-p_i)\right)^2} = \frac{1}{\sum_{i=1}^{n} p_i/(1-p_i)} \leq \frac{1}{p_j/(1-p_j)}$ which is lower, concluding that $w_i = {}^{\alpha}\!/_{(1-p_i)}$ is indeed a minimum.

Let now $V_{\max}$ be the maximum variance admitted by the bounded-loss constraint. Thus, we require that: $V(Y) = \frac{1}{\sum_{i=1}^{n} p_i/(1-p_i)} \leq V_{\max} \Leftrightarrow \sum_{i=1}^{n} p_i/(1-p_i) \geq {}^{1}\!/_{V_{\max}}$. Notice that p_i is the probability of communication succeeding (with the message not being discarded) between the current event ϵ and a corresponding neighbour ϵ_i, which by Equation (8) in Section 3.4 is equal to $p_i = 1 - \text{failingP}(\epsilon, \epsilon_i, \theta) = \text{connectionSuccess}(\epsilon, \epsilon_i) \cdot \text{survivalP}(\epsilon, \epsilon_i, \theta)$. Accordingly, we can choose the maximum threshold granting that the bounded-loss constraint is satisfied with an optimal weight assignment:

$$\theta(\epsilon) = \max\left\{\theta \in [0, \infty] : \ \frac{1}{V(\epsilon, \theta)} = \sum_{\epsilon' \rightsquigarrow \epsilon} \frac{1 - \text{failingP}(\epsilon, \epsilon', \theta)}{\text{failingP}(\epsilon, \epsilon', \theta)} \geq \frac{1}{V_{\max}}\right\} \tag{12}$$

Since $\text{failingP}(\epsilon, \epsilon', \theta)$ is an increasing function of θ, the expression: $\frac{1-\text{failingP}(\epsilon,\epsilon',\theta)}{\text{failingP}(\epsilon,\epsilon',\theta)} = \frac{1}{\text{failingP}(\epsilon,\epsilon',\theta)} - 1$ is a decreasing function of θ. It follows that $\theta(\epsilon)$ can effectively be found by binary search at arbitrary precision.

The overall structure of the C_{blist} algorithm is then similar to C_{list} as described in Equation (7), except for the different criterion for computing $\theta(\epsilon)$ and for the introduction of weights:

$$C_{\text{blist}}(\epsilon) = x(\epsilon) \oplus \bigoplus_{\epsilon' \in E_\epsilon} \left\{w(\epsilon', \delta(\epsilon)) \otimes C_{\text{blist}}(\epsilon') : \ \frac{P(\epsilon') - P(\epsilon)}{t(\epsilon) - t(\epsilon')} \geq \theta(\epsilon')\right\} \tag{13}$$

$$w(\epsilon,\delta) = \begin{cases} \frac{1/\,\text{failingP}(\epsilon,\epsilon_\delta,\theta(\epsilon))}{\sum_{\epsilon' \rightsquigarrow \epsilon} 1/\,\text{failingP}(\epsilon,\epsilon',\theta(\epsilon))} & \text{if exists } \epsilon_\delta \rightsquigarrow \epsilon \text{ on } \delta \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Example 4. *Consider the scenario described in Example 2 and event ϵ_1^3 with suppliers ϵ_1^1, ϵ_1^2. Given θ, we have that: $1/V(\epsilon,\theta) = \frac{1}{\theta/2-0.6} - 1 + \frac{1}{\theta/2-0.05} - 1$. Assume that $V_{max} = 3/59$: then, the highest θ we can choose is $\theta(\epsilon_1^3) = 1.3$. Both following events ϵ_2^1, ϵ_2^2 will then consider the data from ϵ_1^3, since the speed attained for them is $2.2, 1.6 > 1.3$. Their respective weights will be: $\frac{1}{\text{failingP}(\epsilon_1^3,\epsilon_1^1,\theta)} = \frac{1}{\theta/2-0.6} = 20 \;\Rightarrow\; w(\epsilon_1^3,\delta_1) = \frac{20}{65/3} = 12/13$ and $\frac{1}{\text{failingP}(\epsilon_1^3,\epsilon_1^2,\theta)} = \frac{1}{\theta/2-0.05} = \frac{5}{3} \;\Rightarrow\; w(\epsilon_1^3,\delta_2) = \frac{5/3}{65/3} = \frac{1}{13}$.*

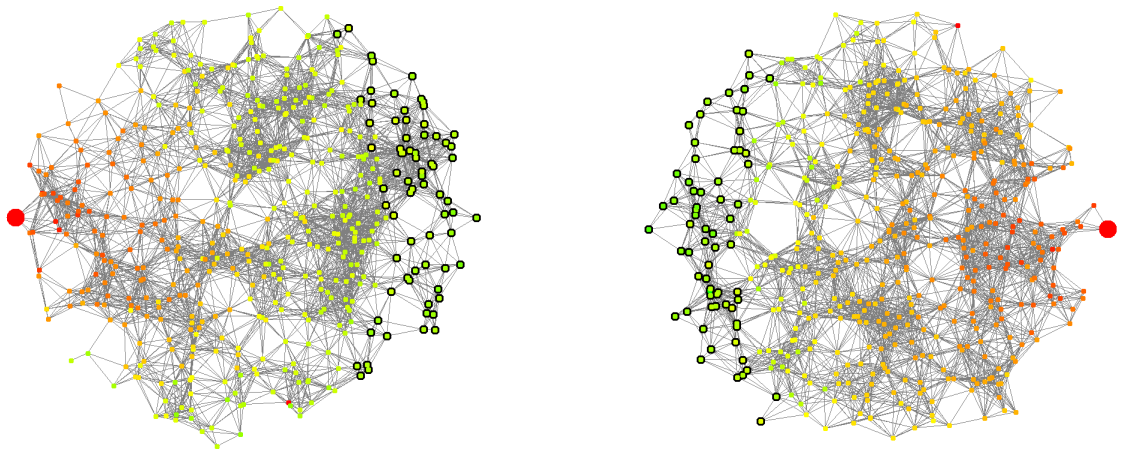
3.7. Resiliency to Network Changes and Self-Stabilisation

The resiliency to network changes of the algorithms presented in this paper can be formally characterised by the notion of *self-stabilisation* [7]. A distributed program is said *self-stabilising* if whenever network structure and input values stabilise, the output values also stabilise (after some time), and the limit output values only depend on the final network structure and input values. This last condition is crucial, as it ensures that after a change, the result does not depend on the values before that change; hence, the algorithm always adjusts to the correct output for the new input. All the algorithms discussed in this paper are self-stabilising, both gradients (G blocks) and collections (C blocks). In particular, all collection algorithms follow the *acyclic* self-stabilising pattern described in [7], where the value of each node in the network is computed as (any) function of the values of nodes with a higher potential. In this way, after potentials reach stabilisation, the node values also stabilise, starting from nodes of maximal potential (farther away from the collector nodes), then proceeding all the way inward until the collector nodes (of potential zero) also stabilise.

Even though self-stabilisation ensures that a correct value is eventually reached in a static situation after a change, it does not pose a limit on how long does it take to reach that final configuration, and it does not apply to mobile situations in which perturbations never stop. In such a scenario, most algorithms exhibit their fragility: single-path aggregation fails to consider large portions of the network, while multi-path aggregation fails to discard obsolete values, leading to double-counting and possibly exponential increase in errors. The C_{list} and C_{blist} algorithms, on the other hand, are designed exactly for these situations, optimising the speed at which obsolete values are discarded, to allow for prompt reactions and avoiding double-counting, while exploiting redundancy whenever possible to reduce (or prevent) data losses.

4. Experimental Evaluation

In this section, we evaluate and compare the proposed algorithms, C_{list} and C_{blist} (cf. Section 3), against state-of-the-art collection algorithms (cf. Section 2), i.e., C_{sp} [7] (single-path collection), C_{mp} [7] (multi-path collection), and C_{wmp} [9] (weighted multi-path collection). Such reference algorithms were already implemented in Protelis [17], a programming language for self-organising systems based on computational



(a) Initially, the collector device (the big red node) is on one side; the devices instructed to vary the values to be collected are on the opposite end, shown with a bold border.

(b) Suddenly, a different collector node is selected at the opposite side of the arena; a corresponding switch also occurs for the devices featuring significant value dynamics.

Figure 2: Evaluation scenario for the idempotent case. The shade of the nodes represents the gradient field: devices closer to the source compute a smaller distance and feature warmer colours with respect to others further.)

fields [7]; so, the proposed algorithms have also been coded in Protelis. We run the evaluation by means of experiments designed to test them especially in stressful conditions. The experiments consist of simulations configured and executed using Alchemist [18], a flexible simulator for networked systems and Protelis applications that has been used in several works [10, 7, 15]. The source code of the algorithms and experiments, the simulation framework, and the instructions for reproducibility are freely available at a public repository[6].

The rest of the section is organised as follows. First, we describe the simulation framework adopted (Section 4.1); then, we discuss the specific experimental configuration and results for two main cases: idempotent aggregation ("minimum", Section 4.2) and arithmetic aggregation ("counting", Section 4.3).

4.1. Simulation framework

Similar experimental scenarios are designed for evaluating both cases (idempotent and arithmetic aggregation). Consider a circular arena filled with many nodes at random positions that compute at a certain, similar frequency and interact with nearby nodes (neighbours). Rounds, which include both computation and broadcasting of data to neighbours, are run once per second by any device; we consider a 10% variance in the rates of rounds in different devices and a further 10% variance rate for a single device. The nodes are mobile devices configured to move randomly in the arena at a fixed, uniform speed. The *sink* device (a.k.a. *collector*—the recipient of the collection process) is initially located at the left end of the arena; then, at

[6] `https://bitbucket.org/roberto-casadei/experiment-optimal-collection`

Param.	Description	Default	Values
hops	Arena diameter as a number of maximal hops	10	[2, 3, ..., 16] (step = 1)
neigh	Average number of neighbours per device	20	[7.0, 10.0,..., 31] (step = 3.0)
speed	Device speed as fraction of *radius* covered in 5s	20	[0.0, 2.5,..., 40] (step = 2.5)
radius	Communication radius in metres	100	

Table 2: Parameters of the simulation scenario

some time instant ($t = 200$ for the arithmetic case, and $t = 300$ for the idempotent case) it switches at the opposite, right end of the arena. Figure 2 shows this reference scenario graphically. Concrete scenarios are produced by varying the following key parameters. *(1)* **Diameter** (*hops*): the circular arena where the nodes are situated has a diameter specified in terms of a number of maximal hops (i.e., multiples of the communication radius). For a fixed communication radius, a higher value of *hops* means that information must cover more space to cover the entire system, since the arena is larger. *(2)* **Average number of neighbours** (*neigh*): this parameter determines the average size of neighbourhoods. For fixed communication radius and diameter, a higher *neigh* means higher average density of devices. Together with *hops*, *neigh* affects the total number of devices considered in a scenario. *(3)* **Speed** (*speed*): the speed at which devices move randomly in the area, expressed as a fraction of the radius of communication covered every 5 simulated seconds. E.g., with a *radius* of 100 metres, a 20% speed means that devices cover 20 metres every 5 seconds, i.e., $4m/s$ (a slow bicycle pace). The concrete values these parameters range over is reported in Table 2. As the resulting matrix of configurations would be huge, we chose to vary each parameter while keeping the others fixed to a default value. The default values for each parameter have been chosen from an extensive exploration of parameter configurations, and have been deemed indicative for the settings of interest. Our goal is addressing *highly mobile and large-scale scenarios*. Addressing huge-scale networks is not a real goal since IoT applications typically feature data locality and divide-et-impera approaches can be used to split a very large area (problem) into multiple areas (smaller problems), with a separate collection process running in each one of them, and then collecting in turn the sub-results of the sub-area collectors. Each distinct configuration is run multiple times by varying a random seed affecting both the actual displacements of devices as well as the simulation dynamics (e.g., the actual timing of round scheduling at the devices). The values obtained for the different runs of each configuration have been averaged to show their tendency. In total, 120 different runs (each of 400 simulated seconds) have been launched, analysed, and reported.

4.2. Idempotent Aggregation

The idempotent aggregation operator in this evaluation is the minimum function, i.e., $\oplus = \min$. So, the value to be collected at the sink is the minimum value among those provided by the devices in the arena. We generate the values yielded by the devices in order to make the task hard for the collection algorithms. For

idempotent aggregation, this means making *obsolete* values produced by *distant* devices *significant*. Indeed, if the values produced in the past are not very significant (e.g., for collection of the minimum, because more recent values are decreasing), then there is little advantage in preferring reactivity, and multi-path collection is preferable. However, if values far from the collector are not very significant (e.g., because values are randomly distributed, or, for collection of the minimum, because small values tend to be concentrated nearby the collector), then data loss has a negligible effect and single-path collection is preferable.

To increase the significance of values far away from the collector, we let the values produced by a group X of devices located at the opposite end of arena vary as per a sinusoid (cf. Figures 2 and 3): $x(\epsilon) = \min(\max(A\cos(2\pi(\min(t(\epsilon), 300) + \phi)/T), -M), M)$ which is a function of the current simulation time instant $t(\epsilon)$, parametrised with amplitude $A = 300$, sinusoid period $T = 250$, phase $\phi = -25$, and projection interval $\pm M = \pm 220$. The other devices, not included in X, are configured to yield irrelevant values (e.g., a constant 400 that is always above the actual minimum). Note that, at $t = 300$, the devices in X and the collector switch to the opposite side of the arena; also, for $t \geq 300$ the inputs $x(\epsilon)$ of the devices in X equal to 220 (the minimum in the network), as per the formula above. This dynamics enables us to inspect the response of the collection algorithms in all the relevant situations: stationary and non-stationary—for both decreasing and increasing values of the true result, and for continuous and discontinuous perturbations.

The experimental results are shown in Figure 3. From them, it is evident that C_{sp} suffers the continuous perturbations induced by mobility. It is confirmed its superior performance over C_{mp} and C_{wmp} in static scenarios (e.g., for *speed* < 5). C_{mp} is effective before $t = 200$, then it takes quite a lot of time to recover. Also, its ability to react suffers significantly as the network diameter *hops* and the density *neigh* raise, while it is relatively uninfluenced by *speed*. C_{wmp} has a rather good performance in the considered scenarios, even though it works better with greater neighbourhood sizes. It is effective especially in very dynamic scenarios, where it outperforms C_{sp} and C_{mp}. Though good, C_{wmp} is dominated by the new proposed algorithms, C_{list} and C_{blist}: these achieve similar performances, with the latter doing better mainly at higher speeds. For *speed* $> 40\%$, C_{list} starts to vacillate a bit, performing like C_{wmp}: this is due to a pessimistic selection of the data speed thresholds, which is bounded to the "zero information lost" constraint. By relaxing this constraint (admitting a very low but not exactly zero chance of information loss) C_{blist} enables data to flow faster, maximising reactivity and hence dealing great with fast mobility. Finally, note in Figure 3 (top left) how the change of the sink at $t = 300$ has only a minor effect on the algorithms' output.

4.3. Arithmetic Aggregation

The arithmetic aggregation operator considered in this evaluation is the sum function, i.e., $\oplus = +$. Every device is instructed to contribute with a value of 1. Given this design, the collector node will accumulate a value estimating the number of nodes included in the entire system. In other words, this defines a decentralised *count* algorithm. Unlike the idempotent case, where a set of devices vary the value they

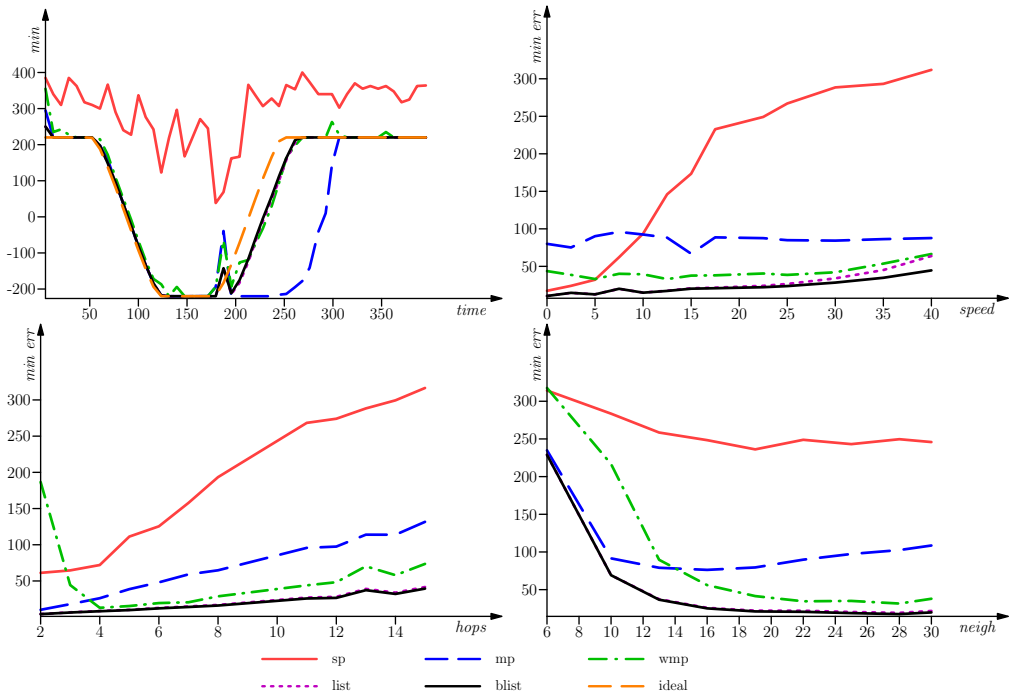


Figure 3: Results of applying the different collection algorithms (C_{sp}, C_{mp}, C_{wmp}, C_{list} and C_{blist}) to the idempotent aggregation case. The top-left plot shows, for the default configuration of parameters (*hops* = 10, *neigh* = 20, *speed* = 20—cf., Table 2), how the value perceived at the sink through the different algorithms relates, over time, to the "ideal" *min* value in the network. The other plots show the average *error* (*min err*) across several runs for varying values of *speed*, *hops* and *neigh*.

provide to the collection, this case has a fixed truth value (corresponding to the total number of devices in the system) for the entire simulation time: the only event injected to perturb the system dynamics is the change of the collector node (as explained in Section 4.1). Note that such a truth value is fixed in a given scenario, but differs across scenarios with different numbers of devices (depending on *hops* and *neighs*).

The results from the experiments can be found in Figure 4. The basic single- and multi-path algorithms C_{sp} and C_{mp} exhibit the poorest performance. The former, C_{sp}, is very good in stationary settings and at very low speeds (0–2.5%), has acceptable performance at moderate speeds (2.5–10%) but tends to severely overestimate (as shown in Figure 4 top left) the true count for speeds higher than 10%. The latter, C_{mp}, on the other hand severely overestimates the true count: the error grows exponentially with higher values of *speed*. Therefore, at the chosen default speed (20%), it collects the worst results—however, the relation with increasing *hops* and *neigh* is similar to that of the other algorithms. Its weighted version, C_{wmp}, does a little better than C_{sp} in moderate to high mobile settings. Except for stationary settings, the algorithms proposed in this paper, C_{list} and C_{blist} provide the best performance. In particular, C_{list} moderately underestimates the true count value. C_{blist} does better than C_{list} especially for high *speed*s, where it exhibits about the 55% of the error of the latter for *speed* = 40%. We considered applying an exponential back-off filter to

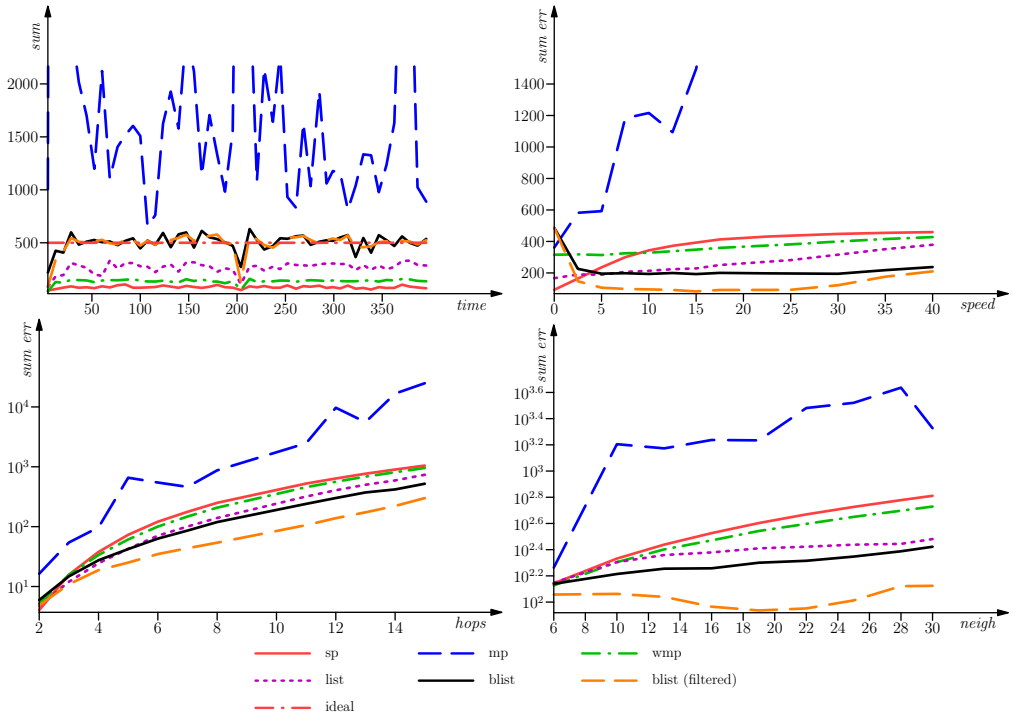


Figure 4: Results of applying the different collection algorithms (C_{sp}, C_{mp}, C_{wmp}, C_{list} and C_{blist}—the latter with and without the exponential back-off filter) to the arithmetic aggregation scenario. The plot at the top left shows, for the default configuration of parameters ($hops = 10$, $neigh = 20$, $speed = 20$—cf., Table 2), how the value perceived at the collector through the different algorithms relates, over time, to the "ideal" *sum* value in the network (i.e., the count of all the devices).The remaining plots (top right, bottom left, and bottom right) show the average *error* (*sum err*) across several runs for varying values of *speed*, *hops* and *neigh*. (High-resolution versions of the plots can be found at the provided repository site.)

provide stability in the aggregation results; from experimentation, we observed a significant positive effect only for C_{blist}. With the use of the filter, C_{blist} further improves its performance, reaching negligible error for medium/small diameters (numbers of *hops*), medium neighbourhood sizes (*neigh* approximately within range $[18, 22]$), and moderate *speed*s (5% to 25%). Finally, notice that when the collector changes (at time $t = 200$), a peak of error is registered for all the algorithms (cf. Figure 4 top left).

5. Case study

To exemplify practical applications, we test the proposed algorithms in a realistic setup of mobile edge computing. We focus on crowd tracking, the process of monitoring the evolution of large assembles of people, with the goal of keeping crowd density under control to prevent dangerous situations. We shall show how such monitoring can be performed fully at the edge of the network. Consider a scenario of crowd tracking in the Italian city of Turin[7], especially pushed after the famous June 2017 incident, when a panic wave and

[7]This would fits Turin initiatives such as the "Smart Square" `https://www.planetsmartcity.com/smartsquare/index.html`

Symbol	Description	Unit	Value
$\vec{v}$	Pedestrian speed	$^{m}/_{s}$	1.6
R	Distance at loss probability 99%	m	50
α	Exec frequency Weibull shape	n.a.	2.5
β	Exec frequency Weibull scale	n.a.	1.2
$^{1}/_{\lambda}$	Mean energy saving delay	s	0.1
$P(d)$	Loss probability	n.a.	(derived)
r	Distance at loss probability 50%	m	$(0.8, 0.85, 0.9, 0.95)R$
C	Device count	devices	$(3, 4, \ldots, 10) \cdot 10^2$
$\mathbb{E}\odot$	Error in crowd barycentre est.	m	(metric)
$P(w)$	Probability of being in danger	n.a.	(metric)

Table 3: Summary of all constants (top), free variables (centre) and metrics (bottom) of the case study, indicating their associated symbol, unit, and (where meaningful) values.

stampede emerged resulting in casualties and injuries [19]. We set up a large event in the popular public park "Parco del Valentino", with an area of roughly $5 \cdot 10^5 m^2$. Suppose a small fraction of participants are equipped with wearable devices given by the organisation, capable of local peer-to-peer communication.

As reference hardware, we model the communication capabilities of the DecaWave DWM1001 Development Board[8][9]. The board features two radio devices, based respectively on Bluetooth Low Energy (BTLE) and on a custom implementation of an Ultrawide Band (UWB) transceiver (IEEE 802.15.4-2011). The BTLE radio is used to exchange identification beacons, while the UWB module is used to communicate directly with devices in proximity, creating a mesh network. The current implementation reaches a communication range of several tenths of meters, and a data rate in the order of few megabits per second. The device is designed to be easily attachable to small-size, low-power ARM boards, such as the Raspberry PI zero, with enough computational power to sustain a full fledged operating system such as Android or GNU/Linux.

In our scenario, nine edge servers are displaced at key locations in the park corresponding to existing facilities. The system goal is to provide edge servers with data for predicting the evolution of the local crowd movement, so that field operators can be informed timely about risky situations. Devices can access their position with reasonable accuracy. Our goal is to estimate the density and barycentre of the local crowd in order to detect risky areas. We run an estimation of the local density, and propagate a warning to devices in potentially risky areas: such data can be used to generate short- and mid-terms density estimates [20].

The case study has been open-sourced for reproducibility, configured with a continuous integration system, and released as DOI objects [21]. Table 3 summarises all the variables, constants, and metrics. To

[8] `https://www.decawave.com/product/dwm1001-development-board/`

[9] This prototype device is priced below €100 and its expected market price is about €20, fully compatible with our scenario.

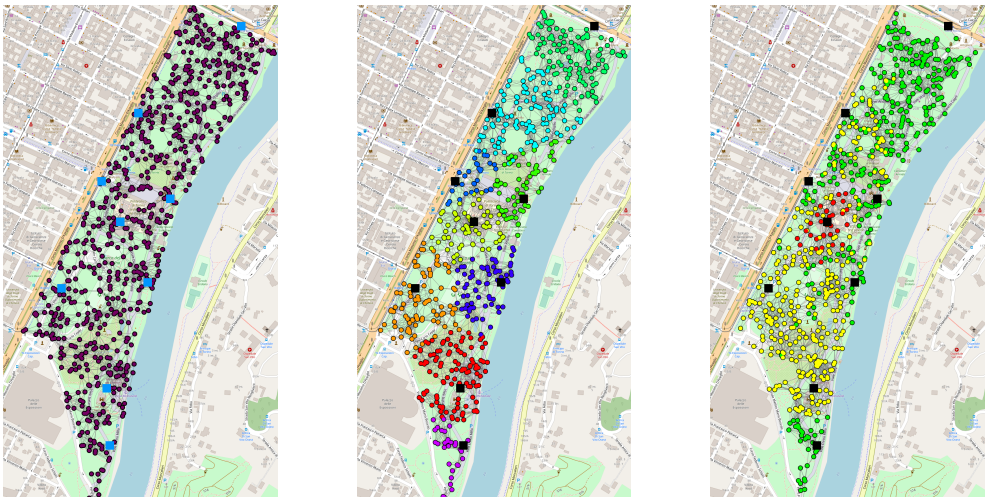


Figure 5: Snapshots of the simulated scenario: *(left)* initial displacement (purple dots are wearable devices, blue ones are edge servers); *(center)* each edge server is given an area (automatically by a gradient), and accumulates and processes data coming from that devices in it; *(right)* final situation (green/yellow/red devices denote areas of low/growing/dangerous density).

challenge our proposed algorithms, we include three elements of realism and a disturbance: *(1)* a packet loss model, with the probability of a packet not being received growing with distance; *(2)* variability in the device working frequency, to emulate unpredicted scheduling policies, e.g., for battery saving; *(3)* people are not stationary, but move towards random waypoints within the park at a speed of $1.6^{m}/_{s}$ [22]; *(4)* once the system initial transient is terminated, we progressively turn off all edge servers, shutting down one edge server every 100 seconds. We measure two parameters: the probability that a device is found in an area where density is growing towards a potentially dangerous level ($P(w)$), and the overall error of the system in identifying the barycentre of the tracked crowd ($\mathbb{E}\bigodot$), compared to the value produced by an oracle.

Our packet loss model builds from an existing model and field experience. We base our model on an existing characterisation of the UWB radio packet loss [23], where distance is found as the most impacting parameter related to signal attenuation, which in turn is linearly related to the probability of packet loss. Loss probability is modelled as a function of distance d between the packet emitter and any receiver. The best fit of the data in [23] was found in a *generalised logistic curve*: $P(d) = A + \frac{K-A}{\sqrt[\nu]{Qe^{-Bd}+C}}$ with parameters $A = 0$, $K = C = 1$ (needed for a upper asymptote 1 and lower asymptote 0) and $\nu = 3$ (derived from the experimental data). We derive the shape and offset parameters Q and D from the more manageable parameters r and R, representing respectively the distance at which 50% and 99% of packets are lost. The resulting function is: $P(d) = (7e^{\log(6\,792\,093/29\,701)\frac{r-d}{R-r}} + 1)^{-1/3}$. To estimate r and R, we measure the behaviour of two DWM1001 devices, obtaining estimate $R = 50m$ and range $(0.8, 0.85, 0.9, 0.95)R$ for r. All results presented in the following sections are the average of simulations executed with these different values for r.

In many situated IoT systems, the operating system likely enforces sleep policies not allowing devices

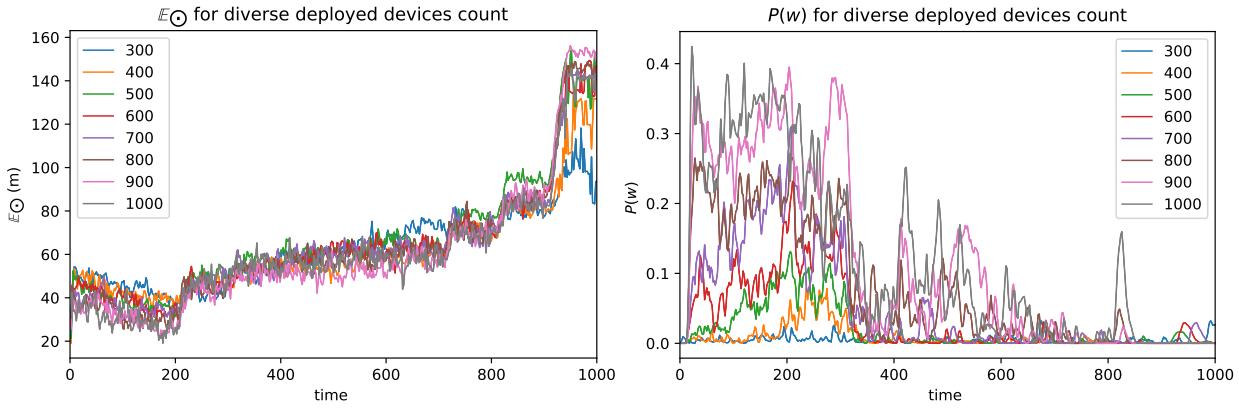


Figure 6: Barycentre estimation error (left) and probability of being notified of a warning situation (right). The system precision is affected by the number of edge servers: less edge servers imply a larger area to be monitored by each. The system is reasonably precise, with an error of few tenths of metres, despite the number of nodes participating in the system. As expected, the more nodes run the system, the higher the chance of such event happening. Nodes are initially displaced randomly, so larger groups form at the beginning of the simulation in the proximity of the park centre as people reach their desired area. With less edge servers, and large areas, the precision of the system decreases sensibly, as results are mediated on a larger area.

to promptly react to received messages or to execute at a constant rate. To model this behaviour, we adopted the following strategy: devices evaluate their program every some time, defined by samples of a Weibull distribution with $\alpha = 2.5$ and $\beta = 1.2$, whose mean is close to 1 and variance close to 0.2. Also, we consider that deep sleep states in microcontrollers may introduce error upon the variability of energy-aware scheduling. As a characterisation of such error is out of the scope of this work, we assumed they cause exponentially distributed delays with $\lambda = 10$Hz (mean delay equal to the standard deviation: $1/\lambda = 0.1$s).

Table 3 summarises all symbols, units, and values for the constants, variables, and metrics of the case study. Snapshots of the simulation execution are depicted in Figure 5, numerical results are framed in Figure 6. All data points are the result of the average over the four values for r described above,and for each combination of values we ran ten repetitions varying the simulator random seed, which influences both the initial displacement of devices in the park and the progression of the simulation. Data shows that the system is able to provide reasonable estimates of the barycentre of the crowd, regardless of the number of participants, and keeping a reasonable quality despite the progressive loss of edge servers. The situation is different in the case of propagated warning: less edge servers mean larger areas, and thus a different estimation of the local density, and hence a lower probability of being warned. However, in real system, a more refined, and possibly learning-enabled strategy, could be devised to propagate warnings only to those devices close to the barycentre of the risky area. Overall, data suggests that the proposed algorithms are indeed applicable in reality, especially in highly dynamic situations where mesh networking is involved.

6. Conclusion

In this article, we introduce two novel algorithms, C_{list} and C_{blist}, addressing distributed collection for both idempotent and arithmetic aggregation. The former is designed to be lossless, the latter to allow a statistical bound on data loss: the key idea of both is to maximise reactivity by regulating information speed through thresholds defined on the basis of a set of network model assumptions. An extensive evaluation is performed to compare these algorithms with state-of-the-art collection algorithms (C_{sp}, C_{mp}, C_{wmp}) in hard conditions characterised by significant levels of scale, density, and mobility. The results witness a marked improvement, providing small errors even in scenarios where devices move fast. Finally, a crowd tracking case study is proposed to show the presented algorithms in action in a significant monitoring application.

Future works can be classified in three groups: *(i)* carrying on a full implementation of the proposed tracking system in collaboration with the Torino City Lab, to experiment with the possibility of providing a set of smart mobility services for Turin citizens, all based on the mobile edge computing paradigm; *(ii)* further improving performance of the proposed algorithms, by using machine learning techniques to optimise various details of system execution, including fire frequency of devices, filtering of neighbours that should receive messages, and selection of algorithm parameters; and *(iii)* considering data collection as a key brick of a full library of adaptive distributed services for mobile edge computing, hence studying new algorithms for other key problems such as gossiping, potential field creating, sparse leader election, and so on.

References

[1] D. Miorandi, S. Sicari, F. D. Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges, Ad Hoc Networks 10 (7) (2012) 1497–1516. doi:10.1016/j.adhoc.2012.02.016.

[2] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J. P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. 98 (2019) 289–330. doi:10.1016/j.sysarc.2019.02.009.

[3] J. Beal, S. Dulman, K. Usbeck, M. Viroli, N. Correll, Organizing the aggregate: Languages for spatial computing, in: Formal and Practical Aspects of Domain-Specific Languages: Recent Developments, IGI Global, 2013, Ch. 16, pp. 436–501. doi:10.4018/978-1-4666-2092-6.ch016.

[4] E. F. Nakamura, A. A. F. Loureiro, A. C. Frery, Information fusion for wireless sensor networks: Methods, models, and classifications, ACM Comput. Surv. 39 (3) (2007) 9. doi:10.1145/1267070.1267073.

[5] F. Dressler, A study of self-organization mechanisms in ad hoc and sensor networks, Comput. Commun. 31 (13) (2008) 3018–3029. doi:10.1016/j.comcom.2008.02.001.

[6] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Networks 52 (12) (2008) 2292–2330. doi:10.1016/j.comnet.2008.04.002.

[7] M. Viroli, G. Audrito, J. Beal, F. Damiani, D. Pianini, Engineering resilient collective adaptive systems by self-stabilisation, ACM Transactions on Modeling and Computer Simulation 28 (2) (2018) 16:1–16:28. doi:10.1145/3177774.

[8] A. Howard, M. J. Mataric, G. S. Sukhatme, Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem, in: Proceedings of DARS 2002, Fukuoka, Japan, June 25-27, 2002, Springer, 2002, pp. 299–308. doi:10.1007/978-4-431-65941-9_30.

[9] G. Audrito, S. Bergamini, F. Damiani, M. Viroli, Effective collective summarisation of distributed data in mobile multi-agent systems, in: Proceedings of AAMAS 2019, Montreal, QC, Canada, May 13-17, 2019, IFAAMAS, 2019, pp. 1618–1626.

[10] J. Beal, D. Pianini, M. Viroli, Aggregate programming for the internet of things, IEEE Computer 48 (9) (2015) 22–30. doi:10.1109/MC.2015.261.

[11] G. Zhou, T. He, S. Krishnamurthy, J. A. Stankovic, Impact of radio irregularity on wireless sensor networks, in: 2nd International Conference on Mobile Systems, Applications, and Services, MobiSys '04, ACM, New York, NY, USA, 2004, pp. 125–138. doi:10.1145/990064.990081.

[12] G. Audrito, F. Damiani, M. Viroli, E. Bini, Distributed real-time shortest-paths computations with the field calculus, in: IEEE Real-Time Systems Symposium (RTSS), IEEE Computer Society, 2018, pp. 23–34. doi:10.1109/RTSS.2018.00013.

[13] Q. Liu, A. Pruteanu, S. Dulman, Gradient-based distance estimation for spatial computers, Comput. J. 56 (12) (2013) 1469–1499. doi:10.1093/comjnl/bxt124.

[14] G. Audrito, R. Casadei, F. Damiani, M. Viroli, Compositional blocks for optimal self-healing gradients, in: Self-Adaptive and Self-Organizing Systems (SASO), IEEE, 2017, pp. 91–100. doi:10.1109/SASO.2017.18.

[15] G. Audrito, F. Damiani, M. Viroli, Optimal single-path information propagation in gradient-based algorithms, Sci. Comput. Program. 166 (2018) 146–166. doi:10.1016/j.scico.2018.06.002.

[16] Y. Zhang, X. Lin, Y. Yuan, M. Kitsuregawa, X. Zhou, J. X. Yu, Duplicate-insensitive order statistics computation over data streams, IEEE Trans. Knowl. Data Eng. 22 (4) (2010) 493–507. doi:10.1109/TKDE.2009.68.

[17] D. Pianini, M. Viroli, J. Beal, Protelis: Practical aggregate programming, in: ACM Symposium on Applied Computing (SAC), 2015, pp. 1846–1853. doi:10.1145/2695664.2695913.

[18] D. Pianini, S. Montagna, M. Viroli, Chemical-oriented simulation of computational systems with ALCHEMIST, J. Simulation 7 (3) (2013) 202–215. doi:10.1057/jos.2012.27.

[19] V. Caramello, L. Bertuzzi, F. Ricceri, U. Albert, G. Maina, A. Boccuzzi, D. C. Francesco, M. C. Schreiber, The mass casualty incident in turin, 2017: A case study of disaster responders' mental health in an italian level I hospital, Disaster Medicine and Public Health Preparedness 13 (5-6) (2019) 880–888.

[20] B. Anzengruber, D. Pianini, J. Nieminen, A. Ferscha, Predicting social density in mass events to prevent crowd disasters, in: Social Informatics - 5th International Conference, SocInfo 2013, Kyoto, Japan, November 25-27, 2013, Proceedings, Vol. 8238 of Lecture Notes in Computer Science, Springer, 2013, pp. 206–215. doi:10.1007/978-3-319-03260-3_18.

[21] D. Pianini, Danysk/experiment-2021-jcee-optimal-converge-cast: Release 0.1.0-2021-02-04t103419 (2021). doi:10.5281/ZENODO.4501063.

[22] R. V. Levine, A. Norenzayan, The pace of life in 31 countries, Journal of Cross-Cultural Psychology 30 (2) (1999) 178–205. doi:10.1177/0022022199030002003.

[23] D. Neuhold, J. F. Schmidt, J. Klaue, D. Schupke, C. Bettstetter, Experimental study of packet loss in a UWB sensor network for aircraft, in: Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems, ACM, 2017, p. 137–142. doi:10.1145/3127540.3127549.

Giorgio Audrito, Ph.D. in Mathematics from the University of Torino, is a research fellow at the Computer Science Department of the same university. His research interests include distributed computing, programming languages, distributed algorithms and graph algorithms. He leads the development of FCPP, a C++-based implementation of the field calculus. Home page: http://giorgio.audrito.info/#!/research.

Roberto Casadei is a research fellow in Computer Science and Engineering at the University of Bologna. With more than 25 publications in international journals and conferences, his research interests and activities

revolve around software engineering and distributed artificial intelligence. He leads the development of the ScaFi aggregate programming toolkit. Home page: `https://robertocasadei.github.io`.

Ferruccio Damiani is an associate professor at the Computer Science Department of the University of Turin. There, he founded and coordinates the System Modelling, Verification and Reuse (MoVeRe) research group, whose goal is to contribute to an effective seamless integration of Formal Methods into software and system development methodologies. Home page: `http://www.di.unito.it/~damiani`.

Danilo Pianini is a research fellow in Computer Science and Engineering at the University of Bologna. He authored over 50 papers journals and conferences on simulation, self-organization, aggregate computing, and software engineering. He is the lead designer of dozens of open-source projects, among which the Alchemist simulator and the Protelis programming language. Home page: `http://www.danilopianini.org`.

Mirko Viroli is Full Professor in Computer Engineering at Alma Mater Studiorum–Università di Bologna (Italy). He is an expert in advanced software development and engineering, author of more than 250 papers (of which more than 70 on international journals). He is member of the Editorial Board of IEEE Software magazine. Home page: `https://www.unibo.it/sitoweb/mirko.viroli`.