Ad Hoc Networks 24 (2015) 29–45

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Data similarity aware dynamic node clustering in wireless sensor networks

Fernando Gielow [a], Gentian Jakllari [b], Michele Nogueira [a], Aldri Santos [a,*]

[a] Department of Informatics, Federal University of Paraná, Brazil
[b] Dept. of Telecommunications and Networks, University of Toulouse, France

ARTICLE INFO

Article history:
Received 26 November 2013
Received in revised form 31 May 2014
Accepted 24 July 2014
Available online 7 August 2014

Keywords:
WSNs
Data similarity
Clustering
Protocols
Bio-inspired

ABSTRACT

Wireless Sensor Networks (WSNs) have been used by several kinds of urban and nature monitoring applications as an important interface between physical and computational environments. Node clustering is a common technique to organize data traffic, reduce communication overhead and enable better network traffic management, improving scalability and energy efficiency. Although current clustering protocols treat various kinds of dynamicity in the network, such as mobility or cluster-head rotations, few solutions consider the readings similarity, which could provide benefits in terms of better use of compression techniques and reactive detection of anomalous events. For maintaining similarity aware clusters, the synchronization of the cluster's average reading would allow a distributed and adaptive operation. In this article, we propose an architecture for dynamic and distributed data-aware clustering, and the Dynamic Data-aware Firefly-based Clustering (DDFC) protocol to handle spatial similarity between node readings. The DDFC operation takes into account the biological principles of fireflies to ensure distributed synchronization of the clusters' similar readings aggregations. DDFC was compared to other protocols and the results demonstrated its capability of maintaining synchronized cluster readings aggregations, thereby enabling nodes to be dynamically clustered according to their readings.

1. Introduction

Even though sensors are increasingly common in practical applications, most of them are primitive, when taking only singular and individual data interpretations into account, without establishing further relations between readings. The most usual relations between data readings are referred to as spatial and temporal relations [1]. For example, for readings such as temperature, humidity and lighting sensor readings are likely similar when taken in regions near each other, due to their **spatial relation**. Similarly, successive readings in a single localization tend to vary gradually due to the **temporal relation**.

When exploring and analyzing data readings collectively one could leverage possible relations in the data readings for building more robust applications. In an urban environment, collectively interpreted data can enable streets traffic analysis so that optimal routes can be determined; spatial patterns of temperature readings can be analyzed for locating heat islands for driving improvements in urban planning; audio readings would determine the level of auditive pollution or even map the sound propagation in the environment, etc.

Wireless Sensor Networks (WSNs) have not reached their maximum potential in term of data collection [2]. They have been used as a **communication interface**

* Corresponding author. Tel.: +55 41 3361 3412; fax: +55 41 3361 3205.
E-mail addresses: fhgielow@inf.ufpr.br (F. Gielow), Gentian.Jakllari@irit.fr (G. Jakllari), michele@inf.ufpr.br (M. Nogueira), aldri@inf.ufpr.br (A. Santos).

http://dx.doi.org/10.1016/j.adhoc.2014.07.008
1570-8705/

between the physical and computational mediums, formed virtually by many data sets [3]. Hence, they are essential for the Cyber-Physical Systems [4], which rely heavily on an interface between the computation and physical environments, and for the advent of the Internet of Things [5], which integrate several kinds of heterogeneous devices that need environment information [6]. Among techniques that aim to provide better network scalability, clustering techniques organize nodes in WSNs into hierarchical logical groups (clusters), allowing data aggregation and organization of the traffic in the network [7]. Further, through the use of clustering techniques, an organization that keeps nodes with similar readings grouped together could bring advantages such as much more efficient data aggregation. Moreover, such similarity aware clusters enhance the capability of detection of anomalous events.

WSNs are dynamic in terms of topology, routes and positioning of the nodes. Thus, clustering mechanisms should be **adaptive** and **reconfigurable**. Nevertheless, little research in protocols that handle simultaneously the correlation and variation of data has been developed up to the moment. Clustering protocols for WSNs may have several objectives: some aim to handle the dynamicity due to mobility [8,9], others try, sometime periodically, to recreate entire cluster hierarchies [10]. However, few protocols consider spatial data similarity [1,11] and even fewer support the dynamic nature of data using a dynamic clustering approach [12]. Thus, the lack of suitable protocols for handling such data highlights the need for a data aware protocol that creates clusters of nodes with **similar readings** in an adaptive and dynamic way.

There are some design difficulties to be handled to provide an adaptive operation that continuously keeps nodes with similar readings grouped together. A readings similarity aware protocol for WSNs should ideally operate in a distributed and self-organizable fashion, avoiding coordination from the sink and complete re-clustering operations. Such characteristics are commonly found in biological systems, whose principles have often inspired distributed networking algorithms [13]. Although biological algorithms have inspired clock synchronization mechanisms [14], they pose an unexplored potential in other kinds of synchronization tasks [15].

In this work, in contrast to our work in [16], where the proposal was still static lacking an Adaptive Agent, we propose a conceptual architecture for dynamic and distributed data-aware clustering, and a logical organization protocol, named DDFC (***D**ynamic **D**ata-aware **F**irefly-based **C**lustering*), that considers spatial data similarity in dynamic environments. The protocol, utilizing the biological principles of fireflies, groups nodes with similar readings. DDFC synchronizes similar reading aggregations in clusters, supporting their **dynamic maintenance** and **internal routing**, thereby enabling an easy detection of nodes which should be clustered together. DDFC acts between the link and network layers, making use of link layer broadcasts to establish logical clusters and perform intra cluster routing. Hence, the network layer uses the clusters created by DDFC, routing messages between the cluster-heads and the sink. Such data similarity aware clusters enable several kinds of applications in the real world. For instance, with seismic data similarity information, patterns can point to eruptions with some weeks of antecedence [17]; with pollution data similarity analysis, water quality could be estimated and this information could be used for identifying possible areas of contamination and emission.

To assess DDFC's general characteristics and capacity of grouping nodes together, simulations were conducted on the Network Simulator, version 3. Using data readings collected from a real environment, DDFC was compared to a variant and another protocol in terms of cluster-heads stability, readings similarity of nodes clustered together and inconsistent routes. Results obtained prove the efficiency of DDFC in keeping nodes with similar data clustered together and in electing adequate cluster-heads.

Our main contributions consist of (i) the usage of the biological principles of fireflies to synchronize atemporal data, different from traditional approaches that employ fireflies to synchronize exclusively temporal based operations or clocks; (ii) a readings similarity aware clustering protocol which differs from other solutions that focus on more static clusters and dynamic indexing, while DDFC focuses on creating and maintaining the clusters dynamically without considering index based network queries; (iii) a thorough evaluation of DDFC through simulations where it is compared to the best scheme available in the literature; the results demonstrate that our scheme improves its performance by being more stable and by decreasing the number of invalid routes.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 presents the principles of fireflies synchronization. Section 4 describes an overview of the data similarity concepts. Section 5 details the DDFC protocol. Section 6 shows its performance evaluation. Finally, Section 7 presents the conclusions and ideas for future work.

2. Related work

WSNs are dynamic in multiple dimensions, such as topology, routes and node locations. Hence, to support the WSN operation, clustering protocols should be adaptive and reconfigurable [18]. Although there are many solutions in the literature for handling several kinds of dynamicities, none performs **data-aware** clustering in a **dynamic** and **scalable** way.

Some protocols handle the dynamicity due to **mobility** by maintaining clusters while nodes arbitrarily transit through the network. The $SPRP_G$ protocol [8], for instance, establishes a spanning tree though a recursive process, in order to establish cluster-heads and gateways to connect the clusters inside the tree. KHOPCA [9] operates proactively through a simple set of rules that defines clusters with variable k-hops. Those rules consider and manipulate a score system, considering a node's neighbors' scores to calculate its own score.

Other approaches support dynamicity through **cluster recreation**, whether periodic or reactive. DCRR [10] considers that clusters are relevant only when there is an event detection, thus being created only on such occasions, while supporting that, continuously maintaining a cluster

index structure is expensive. Similarly, the ESC protocol [19] coordinates the nodes to detect a relevant event so that, with leader node elections based on spatial cells, redundant information is not sent to the base station.

However, despite the rich literature, none of the proposed solutions offers sufficient support for data similarity-aware clustering. Taking **data similarity** into account, the CAG [1] protocol creates on-demand clusters through the flooding of base station generated queries that carry a field informing the acceptable threshold of data readings' differences from nodes that are to be clustered together.

The DACA protocol [20] creates clusters as a query message is forwarded in the network. It aims to reduce energy consumption by eliminating sensor nodes during the result collections. Like CAG, DACH [11] defines readings difference thresholds, creating a virtual hierarchy with several crescent levels of similarity – operation is centralized at the base station.

In the literature, SCCS [12] stands out by establishing dynamic and reconfigurable clusters without needing constant flooding operations, like CAG and DACH. It uses spatial similarity to cluster nodes together and employs compression techniques based on temporal similarity. Its operation is coordinated by the base-station, which determines when the clusters should be recreated.

However, although those protocols are data-aware, none supports clustering in a **dynamic** and **scalable** way. Specifically, CAG depends on constantly flooding the network to establish new static hierarchies and DACA is suitable only for query-driven WSNs. DACH, on the other hand, depends heavily on the base station that collects data from the entire network to effectively establish hierarchies based on a static snapshot of the network. Finally, although SCCS does not depend on constant flooding operations, its maintenance is not suitable as it only allows cluster divisions and still needs a complete re-clustering triggered by the base station.

Thus, a protocol that operates in a more distributed way and establishes dynamic similarity aware clusters is needed for data similarity aware WSNs. A dynamic synchronization operation should be performed, enabling the nodes to be continuously grouped together or split apart, without a complete restructuring triggered by the base station. The biological principles of fireflies [13,15], having inspired several clock synchronization mechanisms, seem to hold an unexplored potential to answer the challenge set forth here.

3. Fireflies synchronization mechanism

WSNs are expected to satisfy properties such as self-organization, fault tolerance, scalability, heterogeneity and decentralization. All these characteristics can actually be found in natural systems. The high dynamicity present in some biological systems is founded on a small set of rules that determine a collaborative behavior, resulting in resources management, tasks scheduling, social differentiation and synchronization – without the need of external control entities [15].

Firefly-based approaches are classified as *bio-inspired systems* [15]. Some species manage to synchronize their fires in a distributed manner. In [21], Mirollo and Strogatz studied the fireflies firing, modeling their behavior through coupled-pulse oscillators. They assume that each firefly has an oscillator which is incremented and gradually synchronizes, as presented in Fig. 1.

Fig. 1 shows the exchange of messages and resulting synchronization of fireflies. Each firefly has an oscillator represented by a vertical line; when its value reaches 1, the firefly will blink, broadcasting a message that triggers the other fireflies' clock adjustments. The repetition of this process for every firefly leads to synchronization, which is specifically illustrated in Fig. 2.

The Fig. 2 shows the synchronization process between the oscillators of two fireflies, called $V1$ and $V2$. Beginning to fire later, the firefly $V2$ has its clock late when compared to $V1$. Thus, in the instant $t1$, the flash of $V1$ makes $V2$ slightly advance its clock. Analogously, with the flash of $V2$ on $t2, V1$ delays its clock. The same situation repeats on the instants $t3$ and $t4$ until the clocks are finally synchronized at $t5$.

Although clock synchronization through oscillator pulses operates in an apparently simple way, it results from the firefly behavior modeling. However, for its use in WSNs, there are characteristics intrinsic to them which are not handled directly. Tyrrell et al. [13] studied how the fireflies oscillators can be applied in wireless ad hoc networks. They showed that several latencies should be considered or even intentionally incorporated by wireless systems, such that their clocks are synchronized in a more efficient way. Among these times are the latency of propagation, transmission, decoding and refraction:

- **Propagation latency** ($T_0^{i,j}$): time demanded for a message to be sent from a given source i to a destination j, proportional to the distance between such nodes;
- **Transmission latency** (T_{T_x}): transmission duration of the synchronization messages. Although in fireflies the message is always the same, independent from the source, wireless networks require message differentiation in order to identify the source. Hence, a synchronization message must be stipulated, being it a standard of pulses or a message preamble, both demanding time for transmission;
- **Decoding latency** (T_{dec}): after receiving the message, there is a period of time required for decoding the headers of each layer as well as the contents of the message themselves;
- **Refraction latency** (T_{refr}): for a higher stability, a refraction period is added after transmitting a pulse, during which no alteration can be performed on a node's local clock.

Ignoring the propagation time and considering the pulsing period T of a firefly, a waiting time T_{wait} is calculated, according to Eq. (1), for the transmission of the synchronization message. The wireless medium requires the waiting time for the better precision of the resulting synchronization between the node clocks. After the pulse transmission, no clock alteration can be performed for the period T_{refr}, which considers the possibility of messages being exchanged in an unpredictable manner, due to the propagation time $T_0^{i,j}$, causing instability of node clocks.

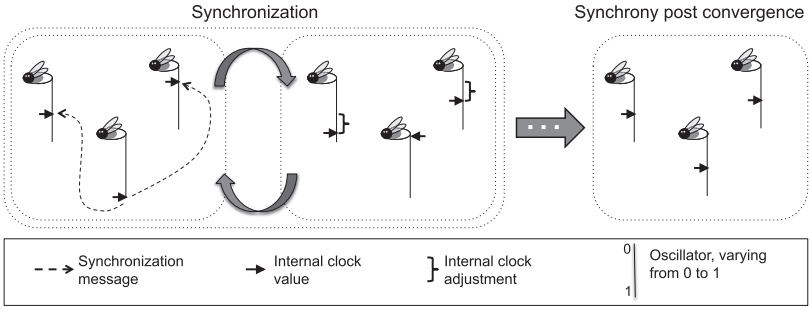

Fig. 1. Fireflies and clock synchronization process.

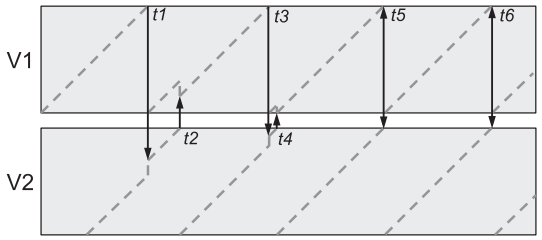

Fig. 2. Couple of firefly oscillators being synchronized.

$$T_{wait} = T - (T_{Tx} + T_{dec}). \quad (1)$$

The RFA (***R**eachback **F**irefly **A**lgorithm*) synchronization mechanism [14] also considers wireless medium latencies. Furthermore, it uses an approach where a firefly waits for its time to pulse in order to perform its clock adjustment, instead of performing it as soon as another firefly pulse is detected. Thus, it enables a clock adjustment only once with the accumulated value and, that way, keeps the mechanism behavior more stable. This operation is specially useful when fireflies have many neighbors and the received pulses adjust the clock alternating between advances and delays, causing minor instabilities. Moreover, such mechanism applies random latencies in the transmission of each pulse, avoiding collisions in the wireless medium.

Similar to RFA, several other works handle synchronization under an exclusively temporal aspect by considering the internal clock of each node [13,22] or focusing on the synchronization of operations based on turns [23,24]. Nevertheless, although the work inspired by fireflies for WSNs focus on the temporal synchronization matter and such temporal synchronization is more evident, it is possible to employ a similar operation to keep the clustering protocol parameters synchronized. In [25], it was shown that although limited to a regional scope, such synchronizations leads also to a global convergence. Hence, the synchronization based on the biological principles of fireflies seems to be appropriate for handling the dynamicity of data readings in a clustering protocol maintenance operation as a whole.

4. Data similarity

There are several approaches in the literature to define WSNs *data similarity*, which depend not only on the handled scope, but mainly on data and on the application. Multimedia applications that handle video streams [26,27] typically associate similarity to (a) overlapping areas in different image frames. After overlapping regions detection, those areas can be easily compressed or even partially eliminated [28].

Regarding scalar data, simple similarity functions can be employed, since they are explicit numeric manipulations. Considering a timeless similarity function, i.e., which is given in a discrete instant and not in a continuous period, common functions commonly involve (b) L absolute difference thresholds between readings a and b, such that $|a-b| < L >$ is satisfied; (c) Q percentage difference between readings, such that b is similar to a if $|a-b| < Q * a >$; (d) customized predefined ranges of readings. Fig. 3 illustrates these similarity concepts.

The similarity of a multimedia frame, Fig. 3a, expresses the region which is common to frames obtained by different cameras. The functions of absolute differences between readings, Fig. 3b, are adequate when there is no necessity for specific ranges of readings to be considered similar and for when the data readings may be expressed by small numerical values. In such cases, the percentage difference, Fig. 3c, clusters the readings in an unequal way: Higher readings will have a larger range of similar readings due to the bigger range generated by the percentage difference. Finally, customized ranges, Fig. 3d, are normally employed when there are predefined distinct groups of interest.

Although these are common approaches, the function to determine the data similarity is strictly dependent on the application and the data. Therefore, for data adequate handling, data similarity aware protocols should ideally adapt their behavior according to a similarity function. Thus, they should allow an easy alteration of the similarity function, without impacting the protocol behavior, which must handle the network dynamicity requirements.

5. Dynamic Data-aware Firefly-based Clustering

Different from our previous work in [16], this section describes in detail the high-level architecture for the DDFC (***D**ynamic **D**ata-aware **F**irefly-based **C**lustering*) protocol, developed to create and maintain logical clustering of nodes that have similar spatial readings. By maintaining local structures for **storing neighborhood** information, DDFC's Firefly Agent synchronizes **local aggregations of**

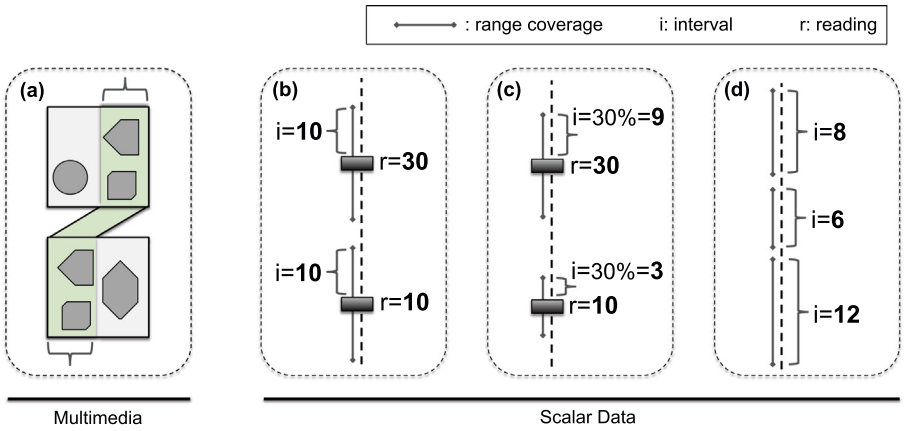

Fig. 3. Examples of data similarity definitions.

similar average readings at nodes, enabling accurate determination of when a cluster should be fragmented or different clusters should be merged together. Once logical clusters are established, DDFC's Indexing Agent defines **indexes for internal routing** on the clusters, enabling the messages from common nodes to reach one of their cluster-heads. In DDFC, clusters can be composed of more than one cluster-head, due to the spatial extent of similar readings. Likewise, the internal routing within a cluster can take more than a hop to reach a cluster-head.

5.1. Overview

The DDFC protocol aims to create and maintain logical clusters of nodes which have similar readings. For that, each node keeps the average aggregation of its cluster readings locally synchronized, in order to verify when a cluster should be fragmented or when different clusters should be merged in the cases of, respectively, readings that do not satisfy or satisfy the desired similarity level. Once having the clusters established, DDFC defines indexes for internal cluster routing, allowing the messages from common nodes to be forwarded to their cluster-head and thus the sink. The general architecture for the protocol is divided in three components, defined as agents, as shown in Fig. 4.

The Firefly Agent is the bioinspired component of the architecture. Its main task is to synchronize the average readings aggregation, thus enabling the cluster maintenance. The Indexing Agent, on the other hand, gives scores for each of the nodes in the network, such that nodes with a given maximum score are taken for cluster-heads, and the remaining nodes use their scores, which are crescent towards a cluster-head, in order to route their data to such cluster-heads. Finally, the Adaptive Agent seeks to dynamically adapt the interval between each beacon broadcast, given that, in conditions of stability, the interval may be increased while in unstable conditions, the interval may be decreased, so that the cluster structures may respond quickly to eventual changes.

Moreover, the general operation of DDFC does not consider energy scarceness to be an issue. Most of the applications envisioned can rely on energy scavenging from the environment, or even fixed sources through electric networks in urban environments, such as street poles. Nevertheless, if DDFC is used by energy constrained applications, several independent energy optimizations can be performed at the link and network layers.

Further, it is important to clarify that the specific kind of dynamicity handled by DDFC is the node **readings dynamicity**. Hence, whenever *dynamicity* is mentioned without a different qualifier, it refers to the data read by nodes, which varies throughout time and space. Analogously, *similarity* and *synchronization* will refer to node data readings and the average synchronization in a neighborhood, as it will be elaborated in the following sections.

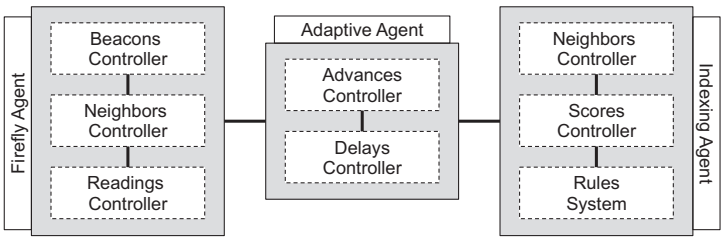

Fig. 4. High level architecture of the DDFC clustering protocol.

5.1.1. Neighborhood information storage

Two simple local structures of great importance in each sensor node support the DDFC operation. Fig. 5 shows a topology and the data structures stored in the nodes, represented by dashed circles. Such data structures correspond respectively to **(i)** information about spatial neighbors data readings and **(ii)** the set of spatial neighbors that satisfy the data similarity thresholds. The highlighted node has seven neighbors, from which, four possess similar readings satisfying the data similarity thresholds.

The set of nodes with similar readings is kept in a structure **(i)** *SNeigh*. Further, a local structure **(ii)** *NeighR* keeps information about all spatial neighbors readings, such information regard individual readings of each neighbor and those neighbors aggregated readings, and the number of nodes whose readings were aggregated.

Initially, each node forms different clusters, which are gradually merged, according to the similarity threshold. After an initial stable formation, the clusters will be maintained dynamically through their union and fragmentation. Algorithm 1 presents the operation of the DDFC's Firefly Agent.

Periodically, each node broadcasts a beacon message, analog to the flashing of a firefly, informing **(i)** its identifier *ADDR*; **(ii)** its current reading, obtained through the function *getReading*(); **(iii)** the average aggregated reading of nodes with similar readings in its neighborhood, obtained through the function *getAverageReading*(), and **(iv)** the quantity of neighbors with similar readings (*l*.1–5). The periodic broadcast of such messages always introduces a random infimum delay in order to avoid simultaneous transmissions (*l*.3).

Algorithm 1. Firefly Agent

```
1:  procedure BeaconTimerExpire
2:    Send(ADDR, getReading(), getAverageReading(), |SNeigh|)
3:    Wait(interval + rnd())
4:    BeaconTimerExpire()
5:  end procedure
6:
7:  procedure ReceiveBeacon(src, iR, aR, nR)
8:    NeighR[src] ← {iR, aR, nR}
9:    localAvg ← getAverageReading()
10:   if (|iR − localAvg| < CThresh)&(|getReading() − aR| < CThresh)
11:     SNeigh ← SNeigh ⋃ {src}
12:   else
13:     SNeigh ← SNeigh − {src}
14:   end if
15: end procedure
16:
17: procedure getAverageReading
18:   accumulatedReading ← getReading()
19:   nOfReadings ← 1
20:   foreach v ∈ SNeigh do
21:     temp ← NeighR[v].aR * NeighR[v].nR
22:     accumulatedReading ← accumulatedReading + temp
23:     nOfReadings ← nOfReadings + NeighR[v].nR
24:   end foreach
25:   return (accumulatedReading/nOfReadings)
26: end procedure
```

5.2. Synchronization of reading aggregations

DDFC defines a synchronization component inspired in the biological principles of fireflies [13], named *Firefly Agent*. That component locally synchronizes a value that indicates the readings aggregation of the current node's cluster. That value enables nodes to know when they should leave their cluster, in the case of readings being too different, and when neighbor clusters should be merged due to similar readings that satisfy the data similarity threshold.

The function *getAverageReading* (*l*.17) calculates the synchronized weighted average of the readings aggregation in the local neighborhood that satisfy the desired similarity, i.e., neighbors which are members of the same cluster. Considering the current node's reading (*l*.18–19), the average of the aggregated readings (*aR*) in the same cluster is calculated (*l*.20–24), using the number *nR* of readings aggregated on that node as a weight (*l*.21). Hence, the average aggregation of similar readings on the region of that node is obtained (*l*.25) and is used to easily repre-

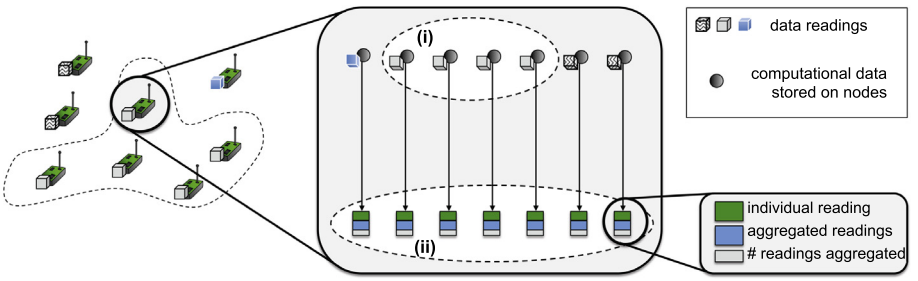

Fig. 5. Data structure for neighborhood information storage: (i) SNeigh and (ii) NeighR.

sent its neighborhood. That way, nodes can check when they should be clustered together.

Upon receiving a beacon (*l*.7), the node will know its origin *src*, the origin's individual reading *iR*, the average aggregated reading *aR* of its neighborhood and the quantity *nR* of nodes whose readings are aggregated. The *NeighR* structure is updated (*l*.8) with such information, independent from any similarity relations – since similarity relations may change, it is important to keep information on every node that may possibly share the same cluster in the future. The average readings aggregation in the region of the current node (*l*.9) is considered to verify determine when whether readings of the current node and the origin node *src* satisfy the data reading similarity threshold *CThresh* (*l*.10). The structure *SNeigh* is then updated, including the origin *src* if the similarity threshold is satisfied – that corresponds to an **union** operation. On the other hand, if the threshold is not satisfied, *src* is removed from such list, corresponding to a **fragmentation**. Hence, such updates of *SNeigh* on both the current and *src* nodes correspond respectively to the union and fragmentation of their clusters, in the cases of similar or different neighborhood readings regarding the desired data similarity threshold.

The similarity function applied on Algorithm 1 consists of two parts: (i) $|iR - localAvg| < CThresh$ and (ii) $|getReading() - aR| < CThresh$, which correspond basically to the same similarity verification, however, with distinct references. The part (i) checks if the reading *iR* received from the neighbor node *src* satisfies the threshold *CThresh*, when compared to the cluster of the current node. On the other hand, part (ii) verifies whether the current reading *getReading*() of the local node satisfies the threshold *CThresh* when compared to the cluster of the node *src*. These two pieces are important in order to guarantee the coherence between what neighbor nodes considers to be similar, i.e., two nodes must agree that they have similar readings bidirectionally.

Fig. 6 illustrates an example of the Firefly Agent's operation, showing the readings aggregation synchronization of each cluster and consequent similarity relations. The dashed edges indicate purely spatial neighbors, while the solid edges indicate neighbors which satisfy the reading similarity threshold. The boxes beside each node correspond to the structure shown in Fig. 5, informing, from top to bottom, the individual reading of that node, the synchronized aggregated reading from it and its neighbors and the quantity of readings that were aggregated there. Each instant T is separated by the broadcast of a beacon from each node. In the initial instant $T1$, the aggregated readings of each node correspond to their own, since no beacon was exchanged yet.

This example considers a value $CThresh = 3.0$, meaning that readings are said similar if their differences satisfy the 3.0 threshold, as previously defined. Hence, the edges $((B,D),(D,C),(C,A))$ satisfy the similarity threshold, establishing similarity relations in the state $T1$. Then, the nodes update their aggregated readings aR_{Tn} according to the earlier instant readings aR_{Tn-1}, as elaborated on the Algorithm 1. In the instant $T2$, $aR_{T2}(A) = \frac{20+1*22}{1+1}$, $aR_{T2}(B) = \frac{26+1*24}{1+1}$, $aR_{T2}(C) = \frac{22+1*20+1*24}{1+1+1}$, $aR_{T2}(D) = \frac{24+1*22+1*26}{1+1+1}$. Thus, the similarity edges (B,C) is created. In the instant $T3$, the aggregated readings are updated again, $aR_{T3}(A) = \frac{21+3*22}{1+3}$, $aR_{T3}(B) = \frac{25+3*24+3*22}{1+3+3}$, $aR_{T3}(C) = \frac{22+2*21+2*25+3*24}{1+2+2+3}$, $aR_{T3}(D) = \frac{24+2*25+3*22}{1+2+3}$. Then, edge (A,B) appears.

This way, each node will have its *SNeigh* structure updated through the exchange of beacons. Such structure indicates which nodes in the neighborhood are seen as members of the same cluster. Thus, as each node knows which neighbors belong to the same cluster, the global

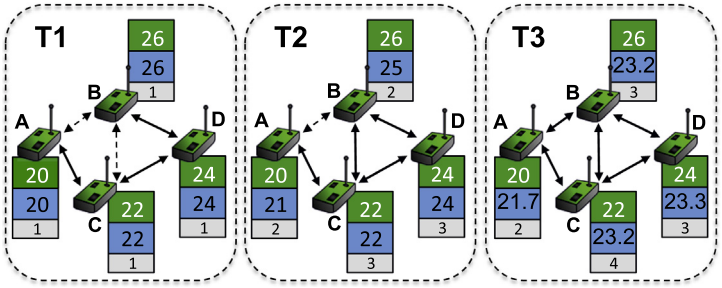

Fig. 6. Firefly agent readings aggregation synchronization on nodes.

cluster of a node corresponds to the set formed by the union of that node with each of the nodes in the *SNeigh* structure. This operation is performed recursively for each node's *SNeigh* structure. Inductively, if a node A belongs to the cluster of a node B and B belongs to the cluster of a node C, then A also belongs to the cluster of node C.

However, this global vision of complete clusters is not maintained locally, as such operation would cause high overhead without the assurance of coherence. Thus, although logical clusters exist globally, their complete formation is not locally visible at the node level. Nevertheless, the knowledge of local neighborhoods allows nodes to establish a distributed cluster-head based hierarchy. With such structures defined by the Firefly Agent, the Indexing Agent, defined in the next section, operates respecting the data similarity, and guarantees that nodes manage to send their messages to a cluster-head in their clusters.

5.3. Indexing of internal routes to cluster-heads

DDFC establishes cluster-heads and routes from common nodes to the nearest cluster-heads through an *Indexing Agent*, that takes the similarity relations established into account. The Indexing Agent uses a Score System, which is based on the rules proposed by KHOPCA (K-HOP Clustering Algorithm) [9]. Through these rules, each node updates its self-attributed score according to the scores of its neighbors in the same cluster – information piggybacked in the beacon used by the Firefly Agent.

A maximum score *MaxK* is defined as a parameter, also determining the maximum distance to a cluster-head. Nodes with a score equal to *MaxK* are elected as cluster-heads, while the remaining nodes use their scores as a way to determine which is the next hop in the routing to the nearest cluster-head. Moreover, such parameter indicates the *MS* maximum time needed for the initial cluster-wide synchronization of the average readings aggregation. As *MS* is directly related to the maximum distance to a cluster-head, it corresponds to $MS = MaxK * int$ in the worst case scenario. However, it should be noted that although *MS* time is needed for the cluster-head to have an average readings aggregation representing the entire cluster, a coherent initial configuration is obtained already with the first beacons, due to the spatial similarity and the neighbor-to-neighbor bidirectional similarity relations.

In the beginning, every node has the same score $pts = 0$. Being *MaxK* the maximum score and *SN*[1] the list of $\boldsymbol{m}$ neighbors of the same cluster, the score *pts* of a node n is updated according to Eq. (2), based on the rules proposed by KHOPCA. The first condition of such equations aims to keep a maximum difference of 1 between the scores of neighbor nodes, given that $MPts(n)$ is the maximum *pts* of nodes in $SN(n)$. The second rule defines a node as a cluster-head, maximizing its *pts* to *MaxK*, in case its neighbors have minimum score. The third rule aims to decrease the score of a node if it has a score greater than all its neighbors, but is not a cluster-head, in order to keep the maximum difference between neighbors score equal to 1. Finally, the fourth rule aims to eliminate the existence of adjacent cluster-heads, nodes with $pts = MaxK$. Such rules compose the Score System.

$$\begin{cases} MPts(n) - 1, & \text{if } pts(m) > pts(n),\ \forall m \in SN(n), \\ MaxK, & \text{if } pts(m) = 0, \forall m \in SN(n), \\ pts(n) - 1, & \text{if } pts(n) \neq MaxK \ \&\ pts(n) > pts(m), \\ & \quad \forall m \in SN(n), \\ pts(n) - 1, & \text{if } pts(n) = MaxK \ \&\ \exists m \in SN(n) \text{ given}, \\ & \quad pts(m) = MaxK \ \&\ ((|SN(m)| > |SN(n)|), \\ & \quad \text{or } (|SN(m)| = |SN(n)| \ \&\ m > n)). \end{cases} \tag{2}$$

This Score System, although based on the rules proposed by KHOPCA [9], was extended to better adapt to the dynamic needs of the environment, showing better stability. The Indexing Agent gives priority, when regarding scores, to those nodes that have more neighbors with similar readings – priority expressed in the fourth rule of the Rules System. Thus, the cluster-heads stability and quality is higher.

These rules are applied periodically at each node, in the same order they were presented, from the first to the fourth, and at each verification, only one rule can be applied. Such operation is different from KHOPCA, which applies more than one rule, resulting in an undetermined order and behavior, which would produce less stable results, as shown in Section 6.

Fig. 7 illustrates the way such rules are applied, using a parameter $MaxK = 3$. Thus, for this example, a common node can be at most 3 hops away from a cluster-head. On the figure, solid edges between each pair of nodes indicate a similarity relation between their readings and, that manner, they consider each other as neighbors, according to the structure *SNeigh*. In the instant $T1$, every node possesses the same minimum score $pts = 0$. In the state $T2$, given that a rules verification does not have synchrony requirements, in our example the nodes B, C and D apply the rules first, maximizing their scores through the second rule. As nodes A and E perform the rules verification later, their neighbors already have $pts = MaxK$ and, thus, A and E apply the first rule. In the instant $T3$, there are three adjacent nodes with $pts = MaxK$. Then, nodes B and D apply the fourth rule, because node C has more similar neighbors. After that, node A applies the first rule again, keeping the difference between adjacent scores as at most 1.

The structure presented in the figure is maintained dynamically in spite of readings and topology variations. With such structure, the nodes that satisfy $pts = MaxK$ are considered cluster-heads. Common nodes can route data to their nearest cluster-head by always forwarding such data to a node that belongs to their *SNeigh* and whose score is greater that their own score. Thus, as the cluster-heads are those with the higher possible score and the rules establish a scores progression towards the cluster-head, it is guaranteed that a cluster-head is always reached on the end of the travelled path.

However, to allow the clusters to adapt more dynamically to readings changes and for node scores to converge more efficiently, it is possible to employ an adaptive

[1] Same *SNeigh* list, abbreviated due to space constraints.

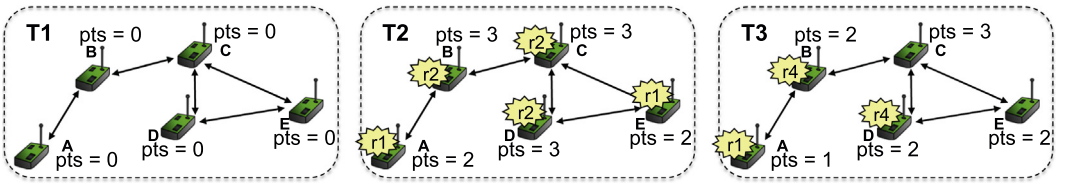

Fig. 7. Indexing agent operation: nodes change their scores based on similar neighbors scores.

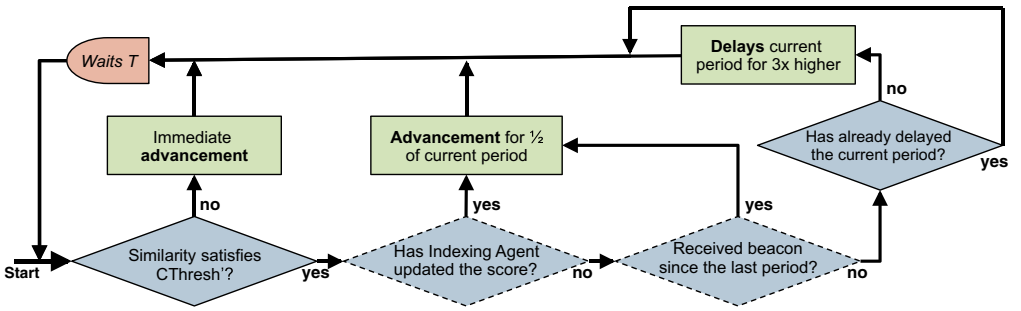

Fig. 8. Operation flowgram of the adaptive agent.

interval between the beacon transmissions. Seeking such behavior, the Adaptive Agent determines how beacon intervals should be delayed or advanced. Such agent is presented in the following section.

5.4. Dynamic adaptation of the actuation intervals

The Adaptive Agent verifies if the Firefly Agent should delay or advance its operation and consequent beacon transmission.[2] Such Agent keeps the similarity information more up to date, so that the Firefly Agent establishes clusters satisfying the reading similarity thresholds more efficiently and the Indexing Agent can update its structure and routes accordingly. Note that, all verifications performed by the Adaptive Agent occur with an interval equivalent to $1/3$ of the default interval of the Firefly Agent beaconing operation, in order to guarantee the existence of time windows sufficiently large between the verifications, so that changes that would have an impact on the cluster formations are more probable.

The Adaptive Agent performs two tests to determine if the Firefly Agent's beaconing operation should be advanced. If the current reading of the node compared to its neighborhood synchronized aggregations does not satisfy a similarity threshold considering a more relaxed $CThresh' = 1.5\ CThresh$, then the beaconing operation is advanced immediately, because the node's reading does not satisfy the clusters aggregation anymore. Here, a larger interval is needed so that small fluctuations do not trigger such advancement. A better value for this relaxed value may be obtained thorough analysis, as it depends on the specific kind of the data being considered by the application.

While the previous condition yielded an immediate advancement, there are less critical conditions that can also result in configuration changes. Such changes may not be present on the current node, but on its surrounding nodes. Hence, if the Indexing Agent has updated the score of the current node in the previous actuation period or if the current node has received a beacon from one of its neighbors in such period, the current period of the beaconing operation should be reduced by half.

With such advancements, stable configurations are established more quickly and, once established, the beacons can be sent with a higher interval. Thus, if no advancement conditions occur, the current period of the beaconing operation should be delayed for three times its current value. The operation of the Adaptive Agent is shown in Fig. 8. Note that, the rhombuses with dashed lines indicate that the condition considers changes since the last verification of the flowgram, which occurs in T time intervals, corresponding to a fraction $1/3$ of the original fixed period of the beaconing operation.[3]

Hence, the Adaptive Agent follows a sequence of verifications which occur in T periods. Although a proactive approach that constantly verifies and adapts the intervals could be used, this periodic verification is preferable because it guarantees a more stable and well behaved operation. As receiving a beacon leads to the advancement of the beaconing period, the proactive operation would be hard to be handled since many beacons may be received in considerably small windows of time. Thus, such verification in a larger time window allows the protocols to perform with more stability and yet, in a dynamic manner.

[2] It must be pointed out that the Indexing Agent acts in the same interval of the Firefly Agent, because the scores information are sent in piggyback in the same beacon.

[3] Discrete intervals are used so that beacons from different nodes may be considered in the same time window. We suggest a default value of $1/3$, but it may be changed according to the application, so that the flowgram is checked with a more appropriate frequency.

5.4.1. Remarks on nodes density and invalid routes

The Adaptive Agent is impacted by the node density in the network. As its operation timer may be advanced on receiving a beacon from a cluster neighbor, it is foreseen that a high density of nodes could cause overhead and instability. Therefore, the use of the Adaptive Agent is adequate for sparser networks. Nevertheless, as the Adaptive Agent is an extension of the Firefly and Indexing Agents, dense networks could still employ the DDFC clustering protocol by disabling the dynamic intervals.

Furthermore, timer advancements generate variations in the timer periods. This decreases the protocol's stability, in order to compensate for faster event detections. Consequently, there is a trade-off: the faster detection depends on a less coordinated exchange of beacons, which may yield temporary invalid routes that are corrected as other neighbors exchange beacons.

Therefore, the usage of the Adaptive Agent depends exclusively on the application needs and network density. By using the Adaptive Agent, an event that would otherwise be detected just after the *int* beaconing interval may be detected up to 66% faster, as a consequence of the lower $int/3$ checking times. Further, although invalid routes appear, they are temporary and always corrected with the exchange of beacons from neighboring nodes.

6. Performance analysis

To evaluate the DDFC protocol performance we implemented it in the NS3 simulator, version 3.14.1. The evaluation scenario creates a realistic environment monitoring application aiming to assess the established clusters efficiency by verifying the readings similarity relations and the quality of the elected cluster-heads. This scenario is based on the humidity readings collected by the Intel Berkley Research Lab [29]. Considering an urban scenario, we assume nodes do not have energy limitations, which could be obtained from existing electric networks, such as trough street poles [30].

The scenario is composed of 54 nodes that operate for 1200 s. As the environment was small, it was amplified in a scale of 15*x*, resulting in a rectangular area of 630 m vs 480 m, given that in the original scenario a standard transmission range would manage to cover all the area, leading to uninteresting results. With this scale, a transmission range of 100 m is used, enabling an evaluation that still has data with spatial relation properties.

Four parameters are varied in the simulations: **(i)** *CThresh*, which indicates the reading similarity among data; **(ii)** *int*, which indicates the fixed interval between beacons; **(iii)** *MaxK*, which determines the maximum possible distance from a common node to a cluster-head, such that the maximum distance is $MaxK + 1$; and **(iv)** *adap*, which indicates whether or not an adaptive interval was considered in the simulation. The ranges $0.5, 1.0, 1.5, 2.0$ were used for *CThresh*; $1, 2, 3$ for *maxK*; and $6, 12, 24, 48, 96$ for *int*.

Apart from that parameters variation for DDFC, another variant of DDFC was implemented, named DDFC-K. Such variation uses the original rules from the KHOPCA [9] protocol on the Indexing Agent, keeping the remaining agents operating the exact same way as DDFC. Such variant aims to verify if the proposed alterations were capable of enhancing the stability and quality of established cluster-heads.

The SCCS (Spatiotemporal Clustering and Compressing Scheme) [12] protocol was also implemented for the evaluation. It was chosen because it presents a clusters maintenance operation which considers the readings similarity to cluster the nodes. The *CThresh* is a common parameter to SCCS, sharing the same meaning as for DDFC. The parameter *int*, for SCCS, is the interval between the transmission of HELLO messages, analog to DDFC's beacons. The main difference between DDFC and SCCS is the need of SCCS to coordinate the network from the base station, determining when clusters should be split apart in order to keep the data similarity threshold satisfied.

The evaluated metrics were: **number of cluster-heads**, **number of clusters**, **number of lone nodes**, **cluster-heads duration**, **average readings amplitude on clusters** and **internal routes inconsistency**. These metrics determine the protocol behavior in how well it dynamically adapts to the readings variation, and the quality of the internal routes from common nodes to cluster-heads it produces.

The number of cluster-heads, clusters and lone nodes are evaluated not only for performance but also to assess the protocol's suitability to possible applications. The cluster-head duration expresses the time a cluster-head manages to keep its maximum score, given that higher average durations indicate that the most suitable cluster-heads were elected. Energy issues are disregarded – the suitability of those nodes in the role of cluster-head is energy independent. The cluster readings amplitude expresses the average difference between the highest and lowest readings in the clusters, and is important for verifying the aggregates synchronization's behavior correctness. Finally, the internal routes inconsistency corresponds to the average number of nodes which cannot reach their cluster-heads with the current network state and node scores.

The results presented in the following were obtained from 35 simulations performed for each parameter combination. The charts present a 95% confidence interval, indicated by vertical bars.

6.1. Established clusters

Fig. 9 presents a set of charts which evaluates the influence of the *CThresh* parameter on the number of cluster-heads, clusters, and lone nodes – i.e., nodes whose cluster consist of only one node. On the left, it can be seen that the higher the *CThresh*, the lower the number of cluster-heads. Further, *MaxK* has the same influence, being more visible between $MaxK = 1$ and $MaxK = 2$. This happens because the higher the *CThresh*, the lower the number of clusters, as seen on the central chart, because more distant readings will be considered similar. *MaxK* acts according to the rules presented in the Indexing Agent, given that higher *MaxK*s yield less cluster-heads. Finally, the number of lone nodes tends to decrease as the *CThresh* parameters

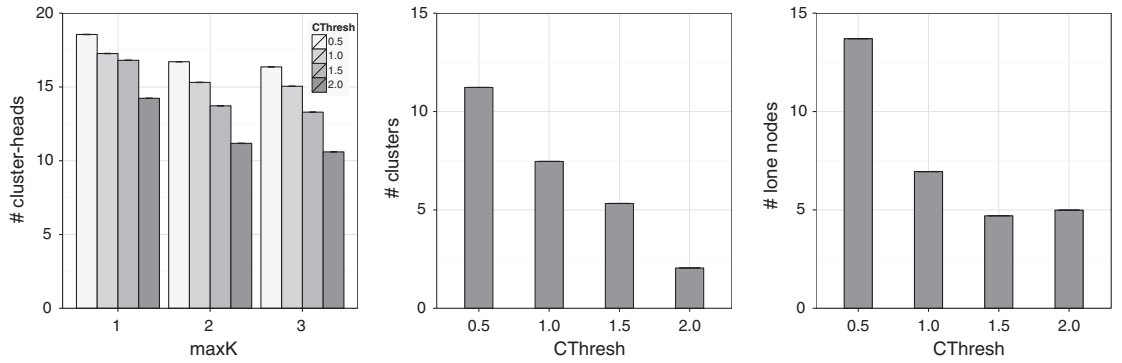

Fig. 9. Number of cluster-heads, clusters and lone nodes.

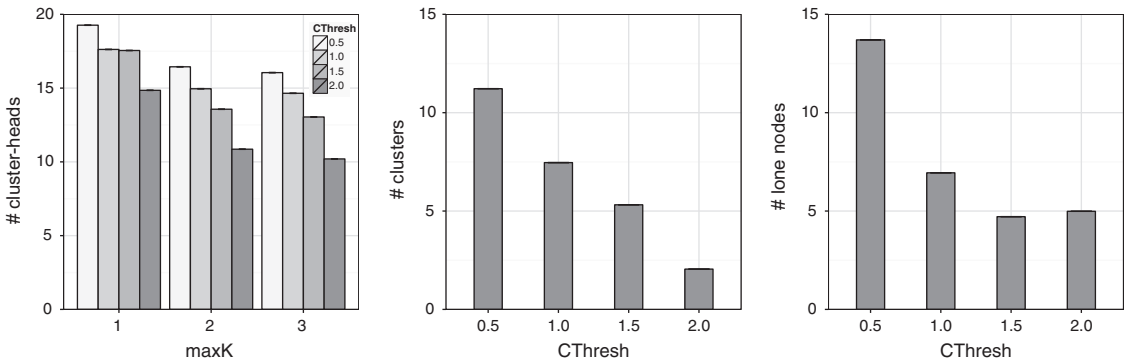

Fig. 10. Behavior of the clusters on DDFC-K.

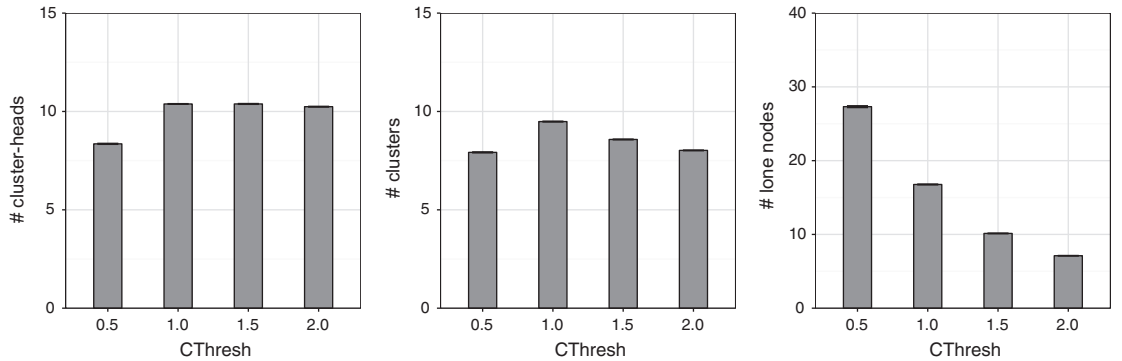

Fig. 11. Behavior of the clusters on SCCS.

increases, given that with higher *CThreshs*, more nodes will be clustered together due to the less strict similarity relation needed. Overall, the average cost in messages exchanged without the Adaptive Agent is $n * int$, where n is the number of nodes.

6.1.1. Comparison with DDFC-K and SCCS

Fig. 10 presents the results obtained for the DDFC-K variation. It can be seen that DDFC-K has results very similar to DDFC. This indicates that the alterations on the rule system have not generated major changes on the network hierarchy as a whole, not being evident on these metrics.

SCCS, on the other hand, showed a very distinct behavior, as seen in Fig. 11. Initially, the number of established cluster-heads is lower, due to the more complex hierarchy of SCCS, which establishes, apart from cluster-heads, gateway nodes to connect adjacent clusters. However, apart from that, the number of cluster-heads and clusters does not follow the *CThresh* growth in a linear way. That happens because the charts present the average values, considering the entire simulation time. With SCCS, there is a tendency for the number of clusters to go up to a certain limit, because the SCCS maintenance consists only of breaking clusters, without any mergers. Hence, even with

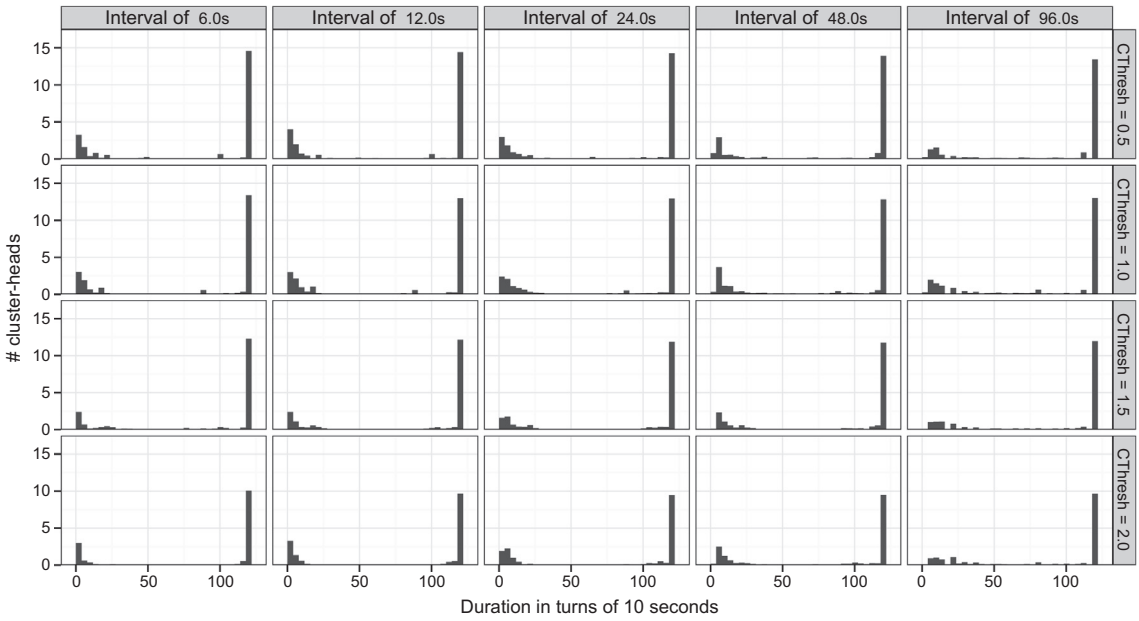

Fig. 12. Cluster-heads duration in turns of 10 s.

a *CThresh* large enough for only a few clusters to exist, they will still be continuously split, without mergers to balance such metric.

The lone nodes metric showed an inadequate behavior with SCCS, given that for small values of *CThresh*, too many nodes in the network are alone in their clusters. This happens because, apart from the clusters breaking before a reclustering process triggered by the sink, the SCCS algorithm does not guarantee that the nodes are consolidated on a permanent state. Many of the nodes were kept in a temporary state *GWR* (candidate to gateway) until the sink would trigger another reclustering process. Furthermore, as it will be shown further, as SCCS does not employ the cluster average readings and only breaks the clusters, the *CThresh* parameters tend to exert a more restrictive influence.

6.2. Cluster-head duration

Fig. 12 presents a set of histograms which relates the number of cluster-heads and their durations, determined in a discrete number of turns, given that a turn represents a time period of 10*s*. The set of histograms is presented in a frame that varies horizontally the parameter *int* and vertically the parameter *CThresh*. It is seen that in every case, higher concentrations of cluster-heads are found for the duration of 120 turns, which in this case corresponds to the entire lifetime of the network. The high duration of these cluster-heads, established according to the Indexing Agent scores, indicates that the used rules establish stable cluster-heads, even with the dynamicity of data readings. This happens mostly because of the fourth rule employed by the Indexing Agent, which was modified to give priority to the cluster-heads with greater number of neighbors with similar readings.

When *CThresh* increases, the number of cluster-heads drops. Although this difference is more pronounces for the maximum duration, it happens in every case, and it happens because a higher *CThresh* yields less cluster-heads in the network as a whole, as seen in Fig. 9. When higher beaconing intervals are used, the duration of the cluster-heads is more dispersed in the intermediate cases, decreasing not only the number of cluster-heads of low duration (i.e., less than 10 turns), but also the number of cluster-heads of maximum duration.

6.2.1. Comparison with DDFC-K and SCCS

Both SSCS and DDFC-K presented lower stability of the established cluster-heads. For 24 s intervals with a $CThresh = 1.0$, for instance, while DDFC presented 13 cluster-heads with maximum duration, DDFC-K and SSCS presented 12 and 9, respectively. Fig. 13 shows the histogram for the cluster-head durations for the DDFC-K variation, which employs KHOPCA's original rules on the Indexing Agent. Comparing such histogram to DDFC's, it can be observed that although *int* and *CThresh* have the same influence on the results, the duration of the cluster-heads for DDFC-K is always inferior, indicating a worse choice of cluster-heads. That happens because DDFC applies modified rules that optimize stability by giving priority to the cluster-heads which have larger neighborhood of similar readings.

Fig. 14 presents the histogram of cluster-heads duration for SCCS. The higher distribution of cluster-heads at lower durations can be immediately observed. SCCS does not manage to establish cluster-heads adequately when *CThresh* is too low because, independent from the established cluster-heads, there is the rupture of the clusters, which is mandatory when the cluster readings diverge, without any further unions. Further, *CThresh* exerts more

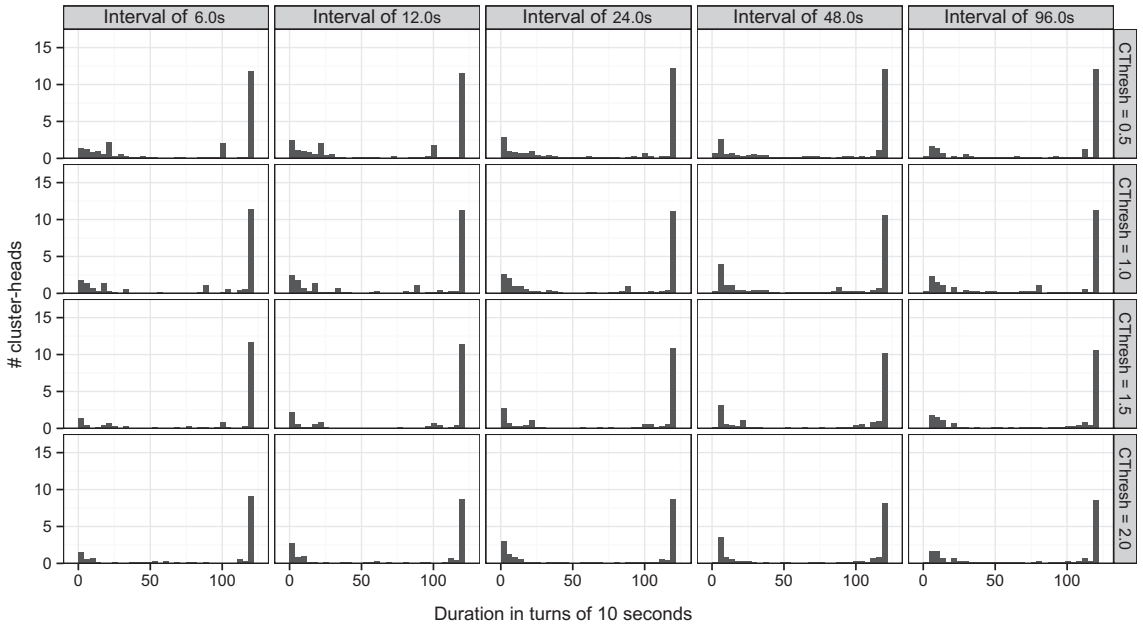

Fig. 13. Duration of cluster-heads for DDFC-K.

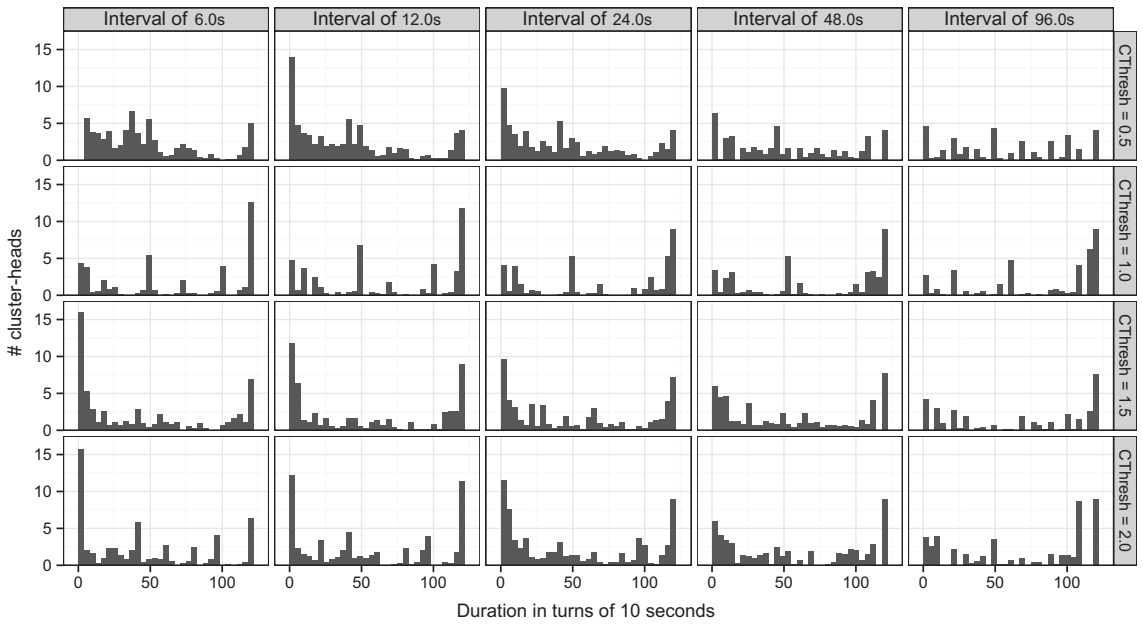

Fig. 14. Duration of cluster-heads for SCCS.

restrictive influence, as will be seen further. However, a more stable behavior is observed for $CThresh = 1.0$, specially for the case of $int = 6.0$. High values for *CThresh* also present instability because the higher these values are, the greater the extension of the formed clusters is leading to a higher likelihood of the cluster to break with the similarity thresholds.

It is hard to establish patterns on the duration of cluster-heads because SCCS behaves in a not deterministic way, depending very much on the order with which the messages are exchanged during the cluster setup phase. Unlike DDFC, which breaks and merges the clusters dynamically in a simple and more efficient way, SCCS can only break them and has to rely on a complete reclustering operation to achieve something analog to the transparent and abstract mergers of DDFC's clusters.

6.3. Similarity of readings among clustered nodes

To determine if the Firefly Agent managed to cluster nodes of similar readings, the amplitude metric is employed, which corresponds to the difference between

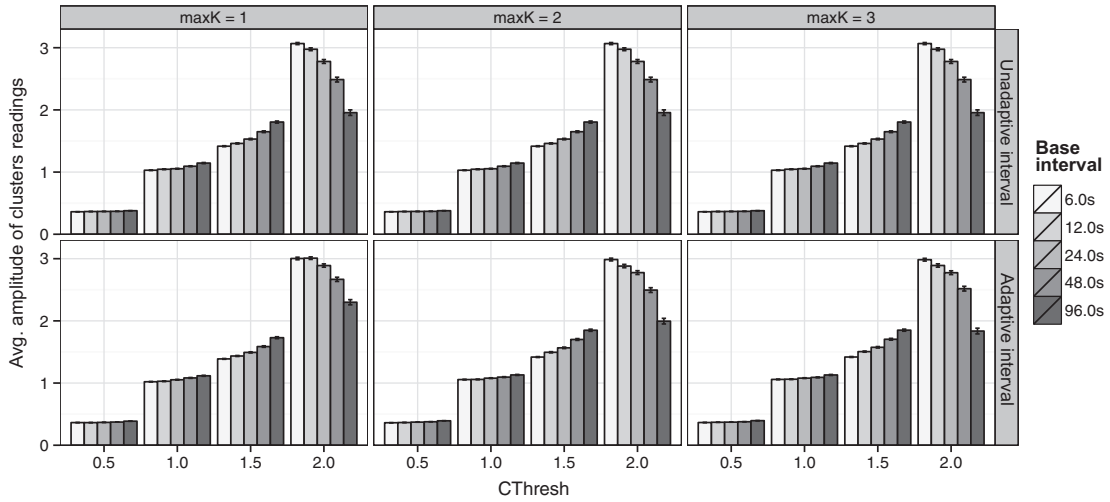

Fig. 15. Similarity of readings inside the same clusters.

the highest and lowest readings in a cluster. Fig. 15 shows the average amplitude of clusters regarding the evaluated parameters. The amplitude increases proportionally to *CThresh*, but always stays below a $2 * CThresh$ limit. That proves that the Firefly Agent has managed to group nodes of similar readings, because given an average reading v, a cluster would accept new nodes in the interval $[v - CThresh, v + CThresh]$, whose amplitude is exactly $2 * CThresh$.

The *MaxK* parameter does not exert much influence on the amplitude of the clusters. Although it influences the number of cluster-heads, as seen in Fig. 9, the number of clusters remains the same, depending only on the similarity threshold and on the *CThresh*. The use of an adaptive interval reduces the amplitude of the clusters, although barely. That happens because although an adaptive approach allows that readings changes and logical clusters become more dynamic and quick, it does not exert much influence on the amplitude itself, given that the raise of dynamicity yields faster adaptation only. Hence, such adaptive approach is beneficial regarding the faster detection of events of interest, as a consequence of the faster clusters formation adaptation to readings variation.

Furthermore as *int* intervals between the beacons increase so does the amplitude of the readings. That happens because with higher intervals nodes will take more time to exchange beacons and thus update the cluster formation to a more coherent state. Moreover, for $CThresh = 2.0$, the behavior of the amplitude variation according to *int* breaks the expected behavior. This happens because, as seen in Fig. 9, the network operates with only two clusters for $CThresh = 2.0$. The network organization in only two clusters is anomalous by itself due to the immense size such clusters reach.

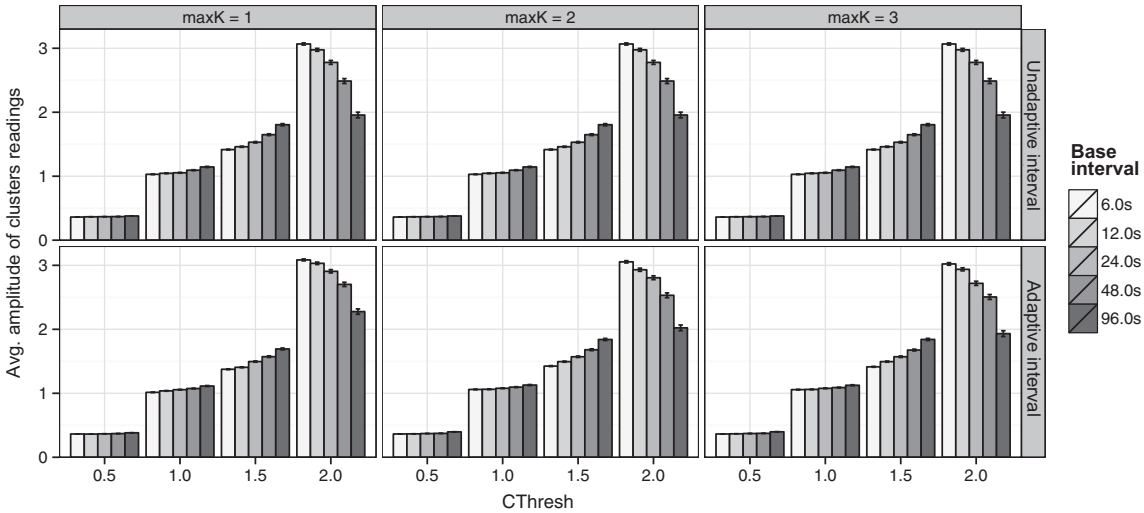

Fig. 16. Readings similarity for DDFC-K.

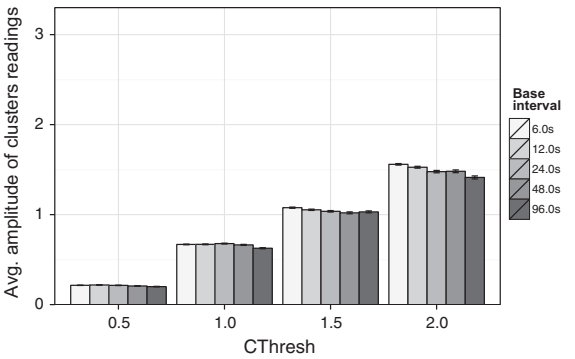

Fig. 17. Readings similarity for SCCS.

6.3.1. Comparison with DDFC-K and SCCS

As seen in Fig. 16, DDFC-K presented amplitudes almost equal to DDFC. With the use of not adaptive intervals, the amplitude found is exactly the same. On the other hand, when adaptive intervals are used, there is a minor variation to the amplitude, though without showing a constant pattern in every case. That indicates such changes occur due to the change of the beaconing timing.

Fig. 17 shows the amplitudes of the clusters established by SCCS. They are always lower, at approximately 50% of the amplitude yielded by DDFC. That happens for two reasons. Initially, DDFC employs the average aggregation of readings in the cluster to compare the similarity among nodes, yielding a flexible behavior in the recognition of new similar readings. On the other hand, SCCS employs always the cluster-head's reading, exhibiting less flexibility. Furthermore, as SCCS can only break its clusters, without dynamically merging them, nodes of the same cluster are in smaller number Fig. 11, contributing to the stricter similarity relation.

Even though this explains the inadequate behavior of SCCS for the case of *CThresh* = 0.5, which generates a practical interval that is too small, in the remaining cases its behavior does not improve significantly. Thus we can conclude that all protocols and variants respected the similarity threshold considered, but SCCS is less flexible and, in practice, considers an interval approximately 50% lower than expected.

6.4. Route inconsistency

Fig. 18 shows the average accumulated inconsistency of routes in the indicated scenarios – i.e., the average of invalid routes throughout all the simulation time. It is noted that the higher the *int* is, the higher the inconsistency. Although an adaptive approach enhances the convergence

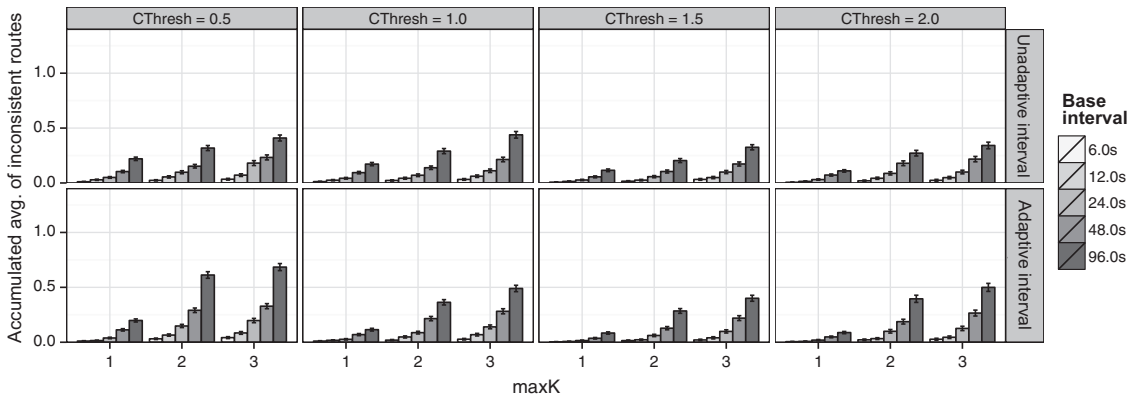

Fig. 18. Accumulated inconsistency of routes.

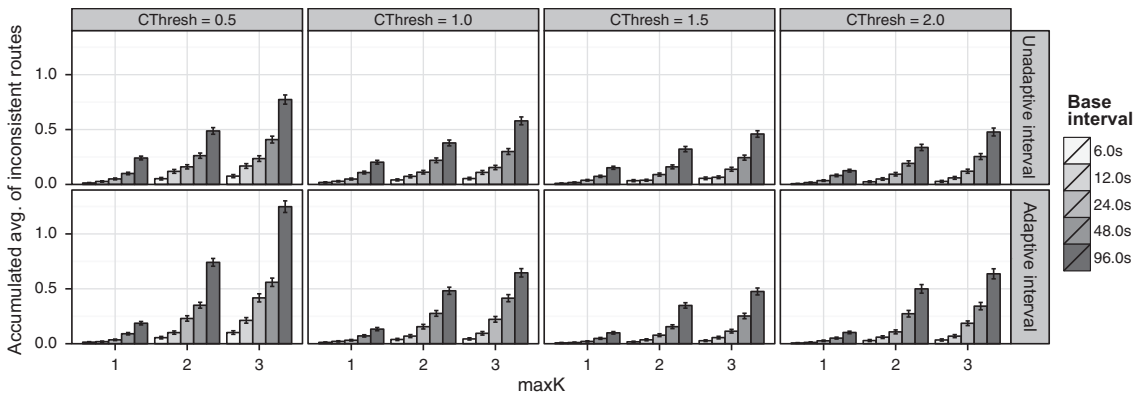

Fig. 19. Inconsistency of routes for DDFC-K.

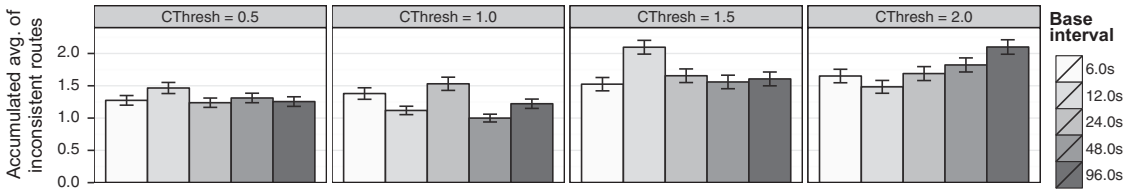

Fig. 20. Inconsistency of routes for SCCS.

time in the start of the network lifetime, it ends generating more invalid routes. Thus, the application should – by itself – determine if the more flexible adaptive detection justifies the slightly larger number of invalid routes. Such Figure also varies the *MaxK* parameter, which increases the number of invalid routes with larger values, due to the greater distances that are made possible between common nodes and cluster-heads.

6.4.1. Comparison with DDFC-K and SCCS

The chart of Fig. 19 shows the inconsistency for the DDFC-K variant. In all scenarios, the proposed modifications on the rule system enhanced the stability of established cluster-heads, by giving priority to the cluster-heads with larger number of neighbors with similar readings. Thus, the inconsistency is smaller for DDFC.

Fig. 20 presents the accumulated inconsistency of routes for SCCS. Two important points are observed. Initially, the inconsistency for SCCS is much superior, corresponding to twice the invalid routes in DDFC, due to SCCS's instability and constant fragmentation of clusters. What is more, there is no obvious pattern between the inconsistency and the *int* parameter. This is because in SCCS there are no clusters unions which explains the insensitivity to the *int* parameter. For DDFC the indexing operation is dynamic and adaptive. Therefore, even though at a given instant there may exist an invalid route, the state of nodes always converge quickly to a valid configuration. In SCCS, there are no route repairs, so one has to rely on the complete reclustering of the entire network.

7. Conclusion

Clusters of nodes with similar spatial readings in WSNs enable more efficient use of aggregation techniques and a more robust detection of anomalous events of interest. Inspired by fireflies, the DDFC protocol employs periodic beacons to keep the readings aggregation synchronized on the nodes of every cluster in an adaptive and reconfigurable approach. Given that, neighbors with similar readings are dynamically identified, enabling the cluster fragmentation and union operations.

The Firefly Agent employs its biological principles in a novel way, differing from the current literature. Meanwhile, the Indexing Agent clusters the nodes dynamically while keeping routing information. An Adaptive Agent was also proposed in order to enhance the former agents further when event detection needs to be performed even faster vis-a-vis the beacon interval.

Such agents maintain clusters of nodes with similar readings, enabling new kinds of applications. In the agriculture, such clusters could be used to adjust water irrigation based on humidity readings from sensors. In the urban environment, clusters of heat and pollution readings can guide health and social projects to enhance the quality of life. Overall, applications that depend on spatial extents similarity information can benefit from using DDFC.

DDFC was evaluated with real readings, obtained from the Intel Berkeley Research Lab. Simulations show that DDFC dynamically keeps the nodes clustered through a synchronized aggregation of the average readings in the clusters, always satisfying the predefined similarity threshold. The rule modifications employed presented better stability, yielding a decrease in the number of inconsistent routes when compared to SCCS. As future work we intend to explore the adaptive control of the interval between each beacon broadcast to decrease the overhead for dense networks.

References

[1] S. Yoon, C. Shahabi, The Clustered AGgregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks, ACM Trans. Sensor Netw. 3 (2007).
[2] C. Partridge, Realizing the future of wireless data communications, Commun. ACM 54 (2011).
[3] Z.-y. Bai, X.-y. Huang, Design and implementation of a cyber physical system for building smart living spaces, Int. J. Distrib. Sensor Netw. (2012) 1–9.
[4] R. Rajkumar, I. Lee, L. Sha, J. Stankovic, Cyber-physical systems: The next computing revolution, in: 47th ACM/IEEE Design Automation Conference (DAC), ACM Request Permissions, 2010, pp. 731–736.
[5] H.-D. Ma, Internet of things: objectives and scientific challenges, J. Comput. Sci. Technol. 26 (2011) 919–924.
[6] T.S. López, D.C. Ranasinghe, M. Harrison, D. Mcfarlane, Adding sense to the Internet of things, Personal Ubiquit. Comput. 16 (2012).
[7] D.J. Dechene, A.E. Jardali, M. Luccini, A. Sauer, A survey of clustering algorithms for wireless sensor networks, Comput. Commun. (2007) 2826–2841.
[8] M. Islam, S. Abdullah, K. Wada, J. Uchida, W. Chen, An efficient routing protocol on a dynamic cluster-based sensor network, in: Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM), 2011, pp. 161–165
[9] M.R. Brust, H. Frey, S. Rothkugel, Dynamic multi-hop clustering for mobile hybrid wireless networks, in: Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, ACM Press, New York, USA, 2008, p. 130.
[10] B. Guo, Z. Li, A dynamic-clustering reactive routing algorithm for wireless sensor networks, Wireless Netw. 15 (2007) 423–430.
[11] X. Wu, P. Wang, W. Wang, B. Shi, Data-Aware Clustering Hierarchy for Wireless Sensor Networks, Springer-Verlag, 2008.
[12] N.D. Pham, T.D. Le, K. Park, H. Choo, SCCS: spatiotemporal clustering and compressing schemes for efficient data collection applications in WSNs, Int. J. Commun. Syst. 23 (2010) 1311–1333.
[13] A. Tyrrell, G. Auer, C. Bettstetter, Fireflies as role models for synchronization in ad hoc networks, in: Proceedings of the 1st International Conference on Bio Inspired Models of Network, Information and Computing Systems, ACM, 2006, pp. 1–7.

[14] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, R. Nagpal, Firefly-inspired sensor network synchronicity with realistic radio effects, in: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, ACM Request Permissions, 2005.
[15] O.B. Akan, F. Dressler, A survey on bio-inspired networking, Comput. Netw.: Int. J. Comput. Telecommun. Netw. 54 (2010) 881–900.
[16] F. Gielow, M. Nogueira, A. Santos, Data similarity aware dynamic nodes clustering for supporting management operations, in: Network Operations and Management Symposium (NOMS), 2014 IEEE, 2014, pp. 1–8, http://dx.doi.org/10.1109/NOMS.2014.6838264.
[17] F. Brenguier, N.M. Shapiro, M. Campillo, V. Ferrazzini, Z. Duputel, O. Coutant, A. Nercessian, Toward forecasting volcanic eruptions using seismic noise, Nature Geosci. (2007) 126–130.
[18] D. Amaxilatis, I. Chatzigiannakis, C. Koninis, A. Pyrgelis, Component based clustering in wireless sensor networks, Comput. Res. Reposit. (CoRR) (2011).
[19] L. Villas, D. Guidoni, R. Araujo, Explorando a correlacao espacial na coleta de dados em redes de sensores sem fio, Simp. Brasileiro Redes Comput. Sist. Distrib. (SBRC 2011) 29 (2011) 411–424.
[20] S. Bahrami, H. Yousefi, A. Movaghar, Daca: data-aware clustering and aggregation in query-driven wireless sensor networks, in: 21st International Conference on Computer Communications and Networks (ICCCN), 2012, pp. 1–7.
[21] R.E. Mirollo, S.H. Strogatz, Synchronization of pulse-coupled biological oscillators, SIAM J. Appl. Math. (1990) 1645–1662.
[22] A. Tyrrell, G. Auer, Imposing a reference timing onto firefly synchronization in wireless networks, in: Proceedings of the 65th Vehicular Technology Conference, 2007, pp. 222–226.
[23] O. Babaoglu, T. Binci, M. Jelasity, A. Montresor, Firefly-inspired heartbeat synchronization in overlay networks, in: Self-Adaptive and Self-Organizing Systems, 2007 SASO, 2007, pp. 77–86.
[24] N. Wakamiya, M. Murata, Synchronization-based data gathering scheme for sensor networks, IEICE Trans. Commun. E88-B (2005) 873–881.
[25] D. Lucarelli, I.-J. Wang, Decentralized synchronization protocols with nearest neighbor communication, in: SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ACM Request Permissions, 2004.
[26] I.F. Akyildiz, T. Melodia, K.R. Chowdhury, Wireless multimedia sensor networks: a survey, IEEE Wireless Commun. 14 (2007) 32–39.
[27] R. Dai, I.F. Akyildiz, A spatial correlation model for visual information in wireless multimedia sensor networks, IEEE Trans. Multimedia 11 (2009) 1148–1159.
[28] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, K. Araki, Practical data compression in wireless sensor networks: a survey, J. Network Comput. Appl. (2011).
[29] Berkeley, 2012. <http://db.csail.mit.edu/labdata/labdata.html>.
[30] S.S. Furlaneto, A.L. dos Santos, C.S. Hara, An efficient data acquisition model for urban sensor networks, in: Network Operations and Management Symposium (NOMS), 2012, pp. 113–120.

Fernando Henrique Gielow is Master in Computer Science by the Federal University of Parana, Brazil. He is a member of the research group NR2, and obtained his B.Sc. in Computer Science by the Federal University of Parana. His research interests include routing and transport on wireless sensor networks, bio inspired approaches and multimedia networks.

Gentian Jakllari was born in Skrapar, Albania. After graduating from the "Petro Nini Luarasi" high school in Tirana, Albania he moved to Greece where he obtained the Bachelor degree in Computer Science from the University of Ioannina in 2002. In 2003 he moved to the USA where he obtained the M.Sc. and Ph.D. degrees in Computer Science from the University of California, Riverside in 2005 and 2007 respectively. From 2007 to 2011 he was a network scientist at the Internetworking Research Department at BBN Technologies in Cambridge, MA, USA. In 2011 he joined INP-ENSEEIHT at the University of Toulouse where he is now an Associate Professor. He holds a joint appointment with the Toulouse Institute for Computer Science Research (IRIT) and is a member of the IRT team. His research interests are in the field of computer networks with emphasis on wireless networks, access layer protocols, routing protocols, cooperative netwoalrks, cognitive radio networks, dynamic spectrum allocation and quality of service provisioning.

Michele Nogueira is Ph.D. in Computer Science by the University of Paris 6, LIP6. Her research interests include security, dependability, network management, performance modeling and wireless networks. She was visiting researcher at the Broadband Wireless Networking Lab, GeorgiaTech, Atlanta (2009) and received her M.Sc. from Federal University of Minas Gerais, Belo Horizonte, Brazil (2004). She was research assistant and consultant on performance analysis of Storage Area Networks (2005). Michele has been a recipient of Academic Scholarships from Brazilian Government throughout her undergraduate and graduate years. She is also member of the IEEE Communication and Information Security Technical Committee, and is involved with the IEEE ComSoc WICE (Women in Communication Engineering) initiative.

Aldri Santos received his Ph.D. in Computer Science from the Federal University of Minas Gerais (Brazil), 2004. Since 2007, he has been an Associate Professor of Department of Informatics at UFPA, leader of the research group in wireless and advanced networks (NR2), CNPq productivity award fellowship PQ2. He is Vice-chair of the Special Interest Group on Information and Computer System Security (CESeg) of the Brazilian Computer Society (SBC). His main research interests are fault tolerance, network management, data dissemination, ad hoc and sensor networks. He has been chair of national and international scientific conferences in the security and management areas.