Distributed Group Consensus Algorithms for Mobile Wireless Sensor Networks

Matthieu Lauzier, Antoine Fraboulet, Jean-Marie Gorce, Tanguy Risset

Distributed Group Consensus Algorithms for Mobile Wireless Sensor Networks

Matthieu Lauzier, Antoine Fraboulet, Jean-Marie Gorce, Tanguy Risset

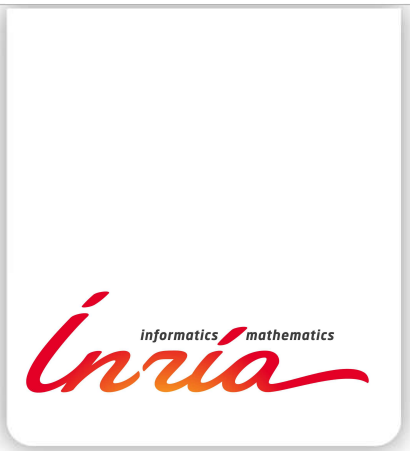
RESEARCH REPORT

N° 8518

April 2014

Project-Teams Socrate

ISSN 0249-6399 ISRN INRIA/RR--8518--FR+ENG



Distributed Group Consensus Algorithms for Mobile Wireless Sensor Networks

Matthieu Lauzier*, Antoine Fraboulet*, Jean-Marie Gorce*, Tanguy Risset*

Project-Teams Socrate

Research Report n° 8518 — April 2014 — 21 pages

Abstract: This paper deals with distributed algorithms for mobile wireless sensor networks, used for monitoring the configuration of a dynamic group. We first present a simple distributed static group consensus algorithm allowing every node in the network to obtain the knowledge of its connected components. We give valid theoretical bounds on the convergence time for this static algorithm. Then we propose two major extensions: the first one consists in using more precision in the proximity information of nodes sharing similar characteristics, the second extension is designed for the case of mobile networks using a periodical reevaluation of the group detection. We validate these algorithms by implementing them in an original and challenging application scenario, in the context of a bicycle race.

Key-words: mobile wireless sensor networks; distributed consensus; topology estimation

* Université de Lyon, INRIA, INSA-Lyon, CITI-INRIA, F-69621, Villeurbanne, France

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Algorithmes de Consensus Distribués pour la Détection de Groupes dans les Réseaux de Capteurs Mobiles

Résumé : Les travaux présentés dans ce document s'inscrivent dans le cadre des algorithmes distribués pour les réseaux de capteurs mobiles, utilisés pour suivre la configuration de groupes dynamiques. Nous présentons dans un premier temps un algorithme simple de consensus distribué appliqué à un réseau statique, grâce auquel chaque noeud du réseau obtient la connaissance de sa composante connexe. Nous fournissons des bornes théoriques valides sur le temps de convergence de cet algorithme. Nous proposons ensuite deux extensions majeures : la première consiste à fournir davantage de précision dans l'information de proximité entre les nœuds partageant des caractéristiques similaires ; la seconde extension est définie dans le cas de réseaux mobiles, et met en œuvre une réévaluation périodique de la détection du groupe. Nous proposons une validation de ces algorithmes par leur implémentation dans le cadre d'un scénario applicatif original, en l'occurrence une course cycliste.

Mots-clés : réseaux de capteurs sans fil mobiles; consensus distribué; estimation de topologie

1 Introduction

Distributed decisions within any group of agents, is a very active research area and theoretical results as well as efficient algorithms have been already proposed [22, 6, 5]. In the context of wireless networks, the task is made harder due to possible transmission errors, channel asymmetry, dynamic behaviour of the channel and node mobility [12, 16].

In this paper, we use this formalism to assess a problem that could be described as a topological problem but which is here interpreted as a consensus problem. We consider a group of mobile agents moving roughly in a common direction and we aim at allowing each agent to discover periodically who are its neighbours in the first vicinity as well as those reachable with multi-hop transmissions. The reference scenario is a bike race, during which groups are susceptible to split or merge. Gathering the information about who is present in a group is the objective. Of course, such an approach may be of interest for various other applications such as group navigation support in crowded environments, autonomous navigation of a fleet of robots...

The objective is to derive a fully distributed approach allowing a group of connected agents to achieve in a finite time a consensus on the list of neighbours participating to the same pack. Considering that each agent contains initially a vector of possible neighbours set to zero except for himself. Then each agent randomly accesses to the medium and transmits its current list to its neighbours. When receiving a list of neighbours, each node eventually updates its values by a OR or MAX operation and prepares to broadcast the updated list of neighbours. This approach is eventually proved to be a straightforward generalisation of a max-consensus algorithm (see [10, 8]), and an upper bound of the maximum delay convergence as a function of the network topology is provided. It is robust to packet losses which maight slightly increase the convergence time. On the opposite, the approach reveals to be extremely sensitive to any false identity transmission that would propagate over the group. We compensate for this issue by adding a strong channel coding to correct and/or detect transmission errors on identity bits.

Last but not least, we extend this approach to a dynamic context where the group knowledge is updated according to possible group gathering or splitting. Unfortunately, the loss of an agent cannot be propagated into a max-based distributed computation. A first possible alternative may rely on introducing a forgetting factor that would allow to remove old information. However, we prefer a more controllable and robust approach that relies on successive epochs at the beginning of which information is reset over the network. Our implementation is fully distributed and doesn't require any global synchronisation. Based again on a max-consensus, a new round is started over the network by propagating a new epoch indicator. All nodes converge to the new epoch at the consensus rate.

Experimental validation has been done in the context of a cycling race, with a sensor located on each bicycle to assess the interactions between the players, i.e. live monitoring of the dynamic evolution of the groups that can be formed during the race. Each bicycle being equipped with a communicating node, the group data aggregation and separation is based on a distance criterion, which takes into account the packet reception rate and received signal strength. We typically want to identify a group separation when the distance between cyclists is greater than 20 meters. Using industrial sensors, we designed the communication protocol and implemented our group consensus algorithm in the bicycle network. We also designed the mobile sink nodes located on the motorbikes surrounding the race that permits the centralisation and exploitation of the data (see Fig. 2). We were finally able to store all the data received by the sensors in order to obtain an accurate database about the behaviour of the network. The designed platform was calibrated for a group of 20 cyclists, experiments were conducted with 10 agents.

The rest of this paper is organised as follows. The next section presents a brief state-of-the-art and highlight our major contributions. The initial algorithm implementing a straight

extension of a max-consensus algorithm is presented in section 3.1 and deals with static nodes (i.e. non-mobile). In section 3.2, two extensions are proposed. While the first one explores the possibility to exchange more than the node identities, the second one deals with mobility. Finally, our solutions are tested on a real experiment. This experiment and the results are presented in section 4 and we provide practical evidence of the efficiency of the Group Consensus Algorithm that we propose.

2 State of the art

As mentioned in the introduction our objective in this work is to achieve a consensus, in each connected component of a graph, about the identity of nodes included in this component. This problem exhibits some similarities with a clustering problem. However, a clustering problem aims at exploiting the structure of the graph to form naturally some subgroups to ensure a good structure of the network for further communications. Typical approaches exploit connectivity and distance measures.

This is why we rather focused on distributed decision algorithms widely present in the literature. In this context, gossip approaches are very appealing [8, 22, 6, 5]. While some works focused on scaling issues to ensure a proper behaviour when the size of the network grows [13], other works focused on the consensus accuracy and convergence speed [22, 6]. In the context of wireless networks, the task is made harder due to possible transmission errors, channel asymmetry, dynamic behaviour of the channel and node mobility [12, 16].

As mentioned in the introduction, our problem can be identified as a max-consensus problem which has been much less studied than average consensus. In [8], the max-consensus is mentioned as one of the possible consensus operation, but the paper doesn't provide specific results about the convergence rate. The most relevant previous contribution is provided by Iutzeler et al. in [10]. It is worth noting that, as mentioned by these authors, the max-consensus problem presents strong similarities with the rumour spreading problem but all previous related works focused either on synchronised networks as done for the Flood-Max algorithm [15] or on pairwise communications (see refs 13-20 in [10]).

Iutzeler et al propose a Random-Broadcast-Max algorithm that relies on 3 steps: i) a node randomly wakes up and ii) transmits its max value. Then iii) neighbour nodes receiving the value update their max values. The algorithm is proved to converge in finite time and the expectation of the convergence time is upper-bounded. They also derived an upper bound for the time convergence holding with a given probability.

In this paper we generalise these results to N-dimensional values. Thus, our algorithm is a vector extension of this max-consensus where the max operation is done component-wise ($v = max(v_1, v_2) \Leftrightarrow v(i) = max(v_1(i), v_2(i))$). We also integrate a reset mechanism that allows to deal with dynamic behaviours and to update the consensus when a group split in independent subgroups.

3 Max consensus algorithms

3.1 Group Consensus Algorithm

In this section, we present the basic version of our algorithm which deals with static nodes meaning that the connectivity between nodes does not change over time.

The static wireless sensor network (WSN) is modelled as an undirected graph $G = (V, E)$. V is the set of sensors and E the set of sensor connections. We denote Δ_G the diameter of graph

G. i.e. the maximum number of edges, between two nodes of the graph. Each link is supposed to be error-free and constant in time without collision: either two sensors can communicate (and in that case, the communication holds in both directions), or they cannot communicate at all. We will release these assumptions in the next section.

The network is asynchronous in the sense that no common clock is available for all nodes. However, local clocks are assumed sufficiently accurate to manage the random transmission clocks. In the random transmission process, each node emits a packet after a random independent and identically distributed (i.i.d) duration t of exponential law of mean λ. It can be shown [22] that, in that case, the process indicating the node number sending a packet at the graph level corresponds to a marked poisson process of density $N\lambda$ whose marks are i.i.d following the uniform discrete law with values $\{1, 2, \ldots, N\}$ These assumptions are commonly used to evaluate consensus algorithms [11, 6, 5].

The Static Group Consensus (SGC) Algorithm aims at allowing each node in a graph to obtain the list of its connected components (i.e. the set of nodes which are connected to him through a multi-hop path. Each node v possesses an internal Boolean N-vector B^v containing the information of the nodes it can reach. After running the SGC algorithm, B^v_i will be 1 if node i and v are in the same connected component. For the sake of clarity, and without lack of generality, the same index in V and B is attributed to each node.

The SGC algorithm is presented in Algorithm 1, Max is the componentwise maximum operation (i.e. boolean 'or' as long as components are boolean) , the ending condition is not straightforward in general. We give below an estimation of the convergence time (section 3.1.1) and we will discuss this point in the practical experiment in section 4.2.2.

Algorithm 1. Static Group Consensus Algorithm

(Executed on each node v)
  Initialize B^v with a 1 at component v and 0 elsewhere
  repeat
    $T_{max} = random(0 - T)$
    While *not expired backoff* T_{max}
      receive(B^j) from node j
      B^v=$Max(B^v, B^j)$
    end While
    broadcast (B^v)
  until *finished*

This algorithm is a straight extension of a Max-Consensus Algorithm [11] for which each node should become aware of the maximal value held by all nodes. While in a Max-Consensus algorithm, each node maximises its own value with the one of its neighbours repeatedly, in SGC each node performs the maximum of its own *vector* with the ones of its neighbours. This extension increases the convergence time and the data transmission load, because in SGC, *all* nodes must send their values (i.e vectors) to all other nodes. However the convergence time analysis performed in in [11] may be extended for SGC.

3.1.1 Performance Analysis

As the graph is static, the algorithm converges on each connected component independently of the others, hence the convergence time can be studied on a single connected component of size N. If the graph G is composed of a single connected component, the convergence is reached when all the B^v are equal to $\mathbf{1}$, hence we are interested in the number of rounds τ needed to

reach the state where all B^v are equal to $\mathbf{1}$:

$$\tau = Min_n\{n \in \mathbb{N} | \forall v \in V, B^v = \mathbf{1}\}$$

Given the fact that the algorithm acts as flooding (the Max operator only keeps the 1 in the B vectors), it is quite obvious that the algorithm ultimately converges and hence that $\tau < \infty$, the following theorem gives an upper bound on the mathematical expectation $\mathbb{E}[\tau]$.

Theorem 3.1. *For the Static Group Consensus Algorithm the convergence time τ is such that $\mathbb{E}[\tau] < N\Delta_G(1 + log(N))$*

Proof. see appendix .1 □

The upper bound obtained in theorem 3.1 is larger than the one of the initial max-consensus algorithm [11] However, the tightness is of the same order, as shown by simulation on randomly generated graphs (section 3.1.2).

Following the approach of [11], we also provide an upper bound on τ with arbitrary high probability.

Theorem 3.2. *For the Static Group Consensus Algorithm the convergence time τ can be bounded with probability $1 - \epsilon$ by:*

$$\tau < N\Delta_G \left(log(N) + log\left(\frac{\Delta_G}{\epsilon}\right)\right)$$

Proof. see appendix .2. □

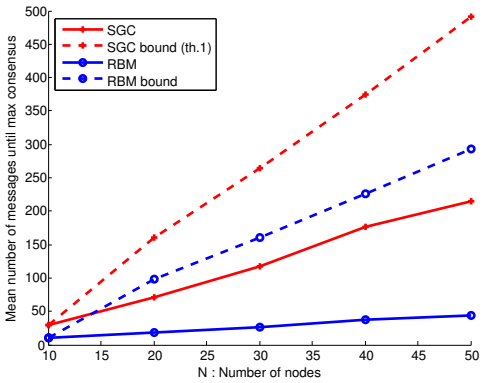
3.1.2 Numerical Simulation

The validity of theorems 3.1 and 3.2 is now illustrated and SGC is compared to the Random Broadcast Max algorithm (RBM), using a similar simulation framework as in [11]. We generated random geometric graphs, composed of N vertices randomly and uniformly placed in the unit square and connected to each other if the distance between them is at most an arbitrary radius r. In this case $r = \sqrt{\frac{8\log(N)}{N}}$, as this value is known for ensuring connectivity with a high probability [20].

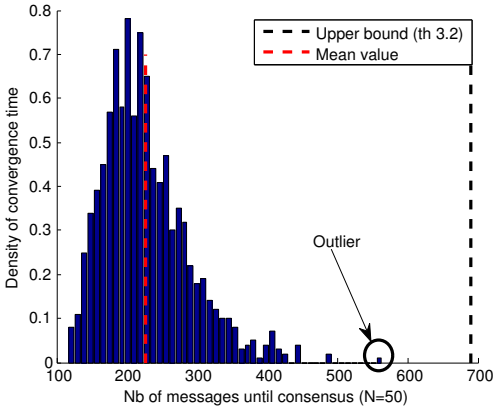
Fig.1a shows a comparison between the mean simulated values obtained for the consensus performance of both RBM and SGC algorithms, as well as their theoretical bounds. Although SGC shows slower performance due to a higher number of communications required to share all nodes information instead of a single maximum value, its performance is close to the upper bound of RBM. In both cases the algorithms converge faster than their upper bounds, and the tightness of the bounds is comparable. It is also interesting to analyse the empirical probability density of the number of communications needed to reach the maximum group consensus. Fig.1b shows the convergence time distribution achieved after 1000 iterations with the SGC algorithm and the associated upper bound given by theorem 3.2 for a probability $\epsilon = 0.1$. Though this distribution is centred around the mean value, it is interesting to observe the apparition of an outlier with a very small probability due to the complete randomness of the transmitting node selection process. Such a measurement is also useful for tuning practical implementations.

3.1.3 Packet transmission errors

Wireless links are known to present relative unreliability, as they are prone to momentaneous disconnections or erroneous received packets. Various factors, depending on the environment, density, communication protocol, have an influence on the communication efficiency. Part of the



(a) Average convergence time versus bound on convergence time expectation for RBM and SGC Algorithms



(b) Convergence time distribution for SGC.

Figure 1: (a) Comparison between algorithms mean performances and their upper bounds for Random Broadcast Max (RBM, [11]) and our algorithm (SGC). (b) Simulation results: density of the SGC convergence time (for 1000 iterations on a $N = 50$ nodes graph), compared to theoretical upper bound on convergence time with probability $1 - \epsilon$, $\epsilon = 0.1$.

work in the consensus community [12] have focused on dealing with erroneous packets in order not to propagate wrong information among all nodes. In the case of transmission failure (collision or transmission errors), it is obvious to that those won't affect the consensus result, but they will have a negative impact on the convergence time. Our broadcast consensus algorithms has nevertheless the advantage of naturally implementing spatial diversity which limits this effect, all nodes acting as communication relays for the others. On the opposite, our algorithm is not robust to faulty packets, since a spurious identifier may eventually proppagate over the whole connected component as if it was a real member. ,This problem is solved in practice by adding a strong channel error detection code. The coding redondancy is indeed used in priority for error detection than error correction, since false positive errors are more critical than non detection errors.

3.2 Mate Group Consensus

The SGC algorithm (Algorithm 1) computes the connected components of the network: nodes aggregate and transmit binary values and finally obtain a vector containing the neighbor list. Depending on the application scenario, it is interesting to extend the information shared by nodes to non binary data (e.g. heat, moisture, pressure, location...). that allows to detect groups according to an application-dependent criterion representing similarity strength among nodes in a given group.

In the context of the platform described in section 4, we propose to transmit an additional information about the adjacent nodes of a group mate as detailed now.

First consider that a node may not take into account the information it receives from a neighbour if a condition between the transmitter and the receiver is not satisfied (e.g. neighbour further than a threshold distance, sporadic connection...) as if the two nodes were not connected. We thus define two nodes being of *mates* if the application condition is satisfied or if two nodes have common mates, to avoid confusion with *neighbours* which refers to nodes able to communicate together (neighbours might not be mates).

Since a node aggregates and transmits the data from its mates, this notion is multi-hop extended, and communities of multi-hop mates are built. This behaviour leads to an artificial separation of the graph into several subgraphs, all composed of multi-hop mates. This approach could be simply implemented with a pure threshold that would remove all badly connected nodes. However, the consensus algorithm is modified to allow the nodes to keep the knowledge about these badly connected nodes.

We present hereafter in Algorithm 2, the Group Mate Consensus Algorithm (GMC), first extension of the SGC algorithm. In section 4.2 we give an explicit example of how the *mate criterion* can be built according to a distance estimation between the nodes using received signal strength indicator (RSSI). The B^v vectors have now integer values (rather than boolean): $B_i^v \in [0; M], M \in \mathbb{N}$, which represent how *close* nodes are with respect to the chosen criteria. When the v node receives a message from node j, it updates its B^v vector if v and j are mates. The set of edges E_C associated to matrix C models the partition of the graph. We also define Δ_C the diameter of the graph given by the adjacency matrix C.

Algorithm 2. GROUP MATE CONSENSUS

(Executed on each node v)
Initialize B^v with M at component v and 0 elsewhere
repeat
$T_{max} = random(0 - T)$
While *not expired backoff* T_{max}

receive(B^j) from node j
proxim=f(RSSI) //proxim between 0 and M
if *(proxim>Threshold) //i.e. j and v are mates*
$B^v = Max(B^v, B^j)$
else
$B^v_j = Max(B^v_j, proxim)$
end If
end While
broadcast (B^v)
until *finished*

3.3 Extension for Node Mobility

When the nodes are mobile, we may have a situation for which a connected component breaks down in two components, or inversely, two independent components merge. Both are referred to as merge-and-split variations. The mobile network can be modelled as a dynamic graph $G(t) = (V, E_C(t))$. The timescale of these variations is assumed relatively large, typically larger than several communication periods. The merging feature is natural with SGC since a new node entering a group is eligible to send its vector which naturally propagates over the group. When a new edge connecting two independent components is created, the vertices of this edge serve as relays to propagate information between the two groups. But, on the contrary, if a node disappears from C_i, the *Max* operation cannot propagate this withdrawal.Dynamically forgetting a node would require to share additional information and to implement complex updating mechanisms with a lack of stability and robustness. We rather introduce a periodical reset of all vectors B^v. This method allows to periodically rebuild the group information, and if a node (or a set of nodes) gets out of a group, the withdrawal appears after the next reset.

The main issue with this approach is the need of a synchronisation method to implement these periodical resets. Let us introduce a global network clock K, called *epoch* and for each node v a local epoch indicator $k_v \in \mathbb{N}$, these clocks are virtual clocks representing an epoch stamps and not to be mistaken with hardware clocks. All local epochs k_v are initialised to 0 and indicate the current *epoch*. These epochs are transmitted with B^v vectors. The synchronisation consists in broacasting a new epoch indicator with a max-consensus algorithm. the initiator may be one of the nodes or an external source.

The initiator triggers a new epoch by incrementing its local clock K and broadcasts it in a *beacon*. All nodes receiving the beacon and for which $k_v < K$, reset their vector B^v and update their local clocks ($k_v = K$). At they turn, they will broadcast a new epoch with a reinitialised B_v. Nodes out of range of the initiator may eventually update their epoch and reset their vector when receiving the new epoch from one of their neighbours. During a certain transition period, some nodes may survive with the epoch $K-1$ while the others already swap to the new epoch. However as the epoch is broadcasted, data of epoch $K-1$ will not be used by nodes having already switched to epoch K. Further, if two non connected components ran for a while with different clocks and evolved to different epochs, in the case they become merged, they will quickly synchronise to the highest epoch.

The Mobile Group Mate Consensus Algorithm (MGMC) is described in Algorithm 3, with external synchronisation.

Algorithm 3. MOBILE GROUP MATE CONSENSUS

(Executed on each node v)
Initialize B^v with M at component v and 0 elsewhere

Initialize k_v to 0
repeat *infinitely*
  $T_{max} = random(0 - T)$
  While *not expired backoff* T_{max}
    receive packet P
    If $P = \{k_j, B^j\}$ *from node j*
      *proxim=f(*RSSI*) // proxim between 0 and M*
      If $k_j > k_v$ *//change epoch*
        $k_v = k_j$
        set B^v *to its initial value*
      end If
      If *(proxim>Threshold) //i.e. j and v are mates*
        $B^v = Max(B^v, B^j))$
      else
        $B^v_j = Max(B^v_j, proxim)$
      end If
    If *P is beacon with epoch K*
      If $K > k_v$ *//change epoch*
        $k_v = K$
        set B^v *to its initial value*
      end If
    end If
  end While
  broadcast $(\{k_v, B^v\})$
until

4 Experiments

In this section we present an original application scenario which allowed us to implement and evaluate the performance of MGMC under real mobile conditions and strong communication constraints. We designed a cycling race wireless sensor network, for assessing interactions between the riders and monitor the groups that can be formed during the race. A group is considered to split when the distance between the cyclists becomes greater than 20 meters.

As far as we know, very few studies have focused on bicycle sensor networks. For instance, the BikeNet system [9] aims at designing a network that collects various information about the individual cycling experience rather than the inter-bicycle communications. The empirical study conducted in [7] shows that the human body has a strong negative impact on the transmissions at the frequencies around 2.4 GHz. Also considering studies performed on Body Area Networks [14] at these frequencies, as our sensors are located in the vicinity of human bodies, we can expect an important attenuation due to the cyclists themselves, as well as very strong and fast variations on the links' qualities, also due to the multipath fading and environment evolution. We need to be aware of the topology changes at the timescale of the group motion, i.e. detect group splitting or merging within a few seconds. In addition, we have to take into account several mobile sinks which may appear or disappear in an unknown manner, and we expect a refresh rate inferior to 1 Hz.

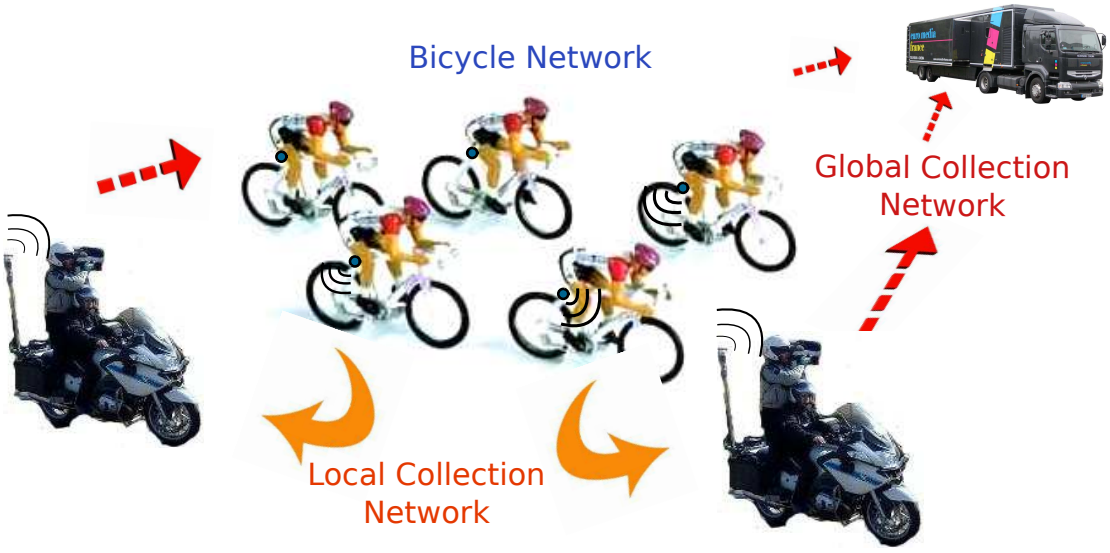


Figure 2: Global infrastructure of the developed platform, illustrating the 3 data collection levels of the network.

4.1 Experimental setup

The whole network infrastructure illustrated in Fig.2 is composed of three levels. The *Bicycle Network* refers to the wireless sensor network formed by the nodes located on the bicycles; the *Local Collection Network* located on motorbikes surrounding the race aims at collecting the data shared by the *Bicycle Network* and acts as a gateway, transmitting the collected information through a dedicated *Global Collection Network* to a central sink located on a truck, where the data exploitation is performed. In this section we describe in details both the *Bicycle Network* and the *Local Collection Network*, but not the *Global Collection Network*, which can be considered as a long-range RF tunnel, since the central point is located far away from the event.

4.1.1 Bicycle Network

Each bicycle is equipped with a HiKoB FOX sensor [2], fixed under the saddle as imposed for the cyclists' comfort. These sensors embed an Atmel AT86RF231 radio chipset embedding a IEEE 802.15.4 compliant PHY layer in the 2.4GHz ISM band [1]; the integrated processor, used for the implementation of application algorithms and communication protocols, is a 32bits ARM Cortex M3 processor. As required for mobility, the FOX sensors run on batteries and embed a micro-SD storage facility, offering several hours of autonomy and storage capacity.

The radio chipset gives access to an Energy Detection (ED) measurement, which is an average of the instantaneous Received Signal Strength Indication (RSSI) over the last 8 modulation symbols received, with a 1dBresolution, from -91dBm to -8dBm. In all the experiments the transmission power will be set to the maximum (3 dBm), to exploit the wider ED measurement range. Table 1 summarises the important radio features of the FOX sensors.

Bitrate	250kbps
Modulation	Offset-QPSK with spreading
Frequency range	2408-2480 MHz
Inter-channel space	5 MHz
Max. transmission power	3 dBm (2 mW) ± **3 dBm**
Sensitivity	-101 dBm at 250 kbps
Energy Detection (ED) range	[-91;-8] dBm ± 1 dBm
Antenna	Chip (integrated on the PCB)

Table 1: Main radio features of the FOX sensors.

4.1.2 Local Collection Network

The system located on each motorbike is a HiKoB LION router [3], which embeds a processor from the same family as the FOX sensor, and the same AT86RF231 radio chipset. It is connected to an external, high gain antenna, and directly powered on the motorbike. The received data is then formatted and transmitted through a USART on a standard asynchronous RS232 serial link with a 9600 bps bitrate before entering the Global Collection Network RF tunnel.

4.2 Calibration

Before the real race, preliminary experiments have been performed to build a rough distance estimator and to adapt and validate the communicaiton protocols.

4.2.1 Empirical distance evaluation

According to the experimental scenario, every node must be able to estimate a rough distance with its 1-hop neighbours. Various methods can be found in the literature for evaluating the distance between two sensor nodes, both hardware and software-based, using either GPS,D-GPS, Angle of Arrival, Time of Arrival, RSSI, and algorithms such as multilateration [4, 18, 21, 19], with various technologies, accuracy and energy consumption. Dedicated hardware extensions for localisation or distance estimation, in spite of a better accuracy, are usually costly in terms of energy or form factor, and many works envetually turn out to consider the Received Signal Strength Indicator (RSSI), which is directly available on the 802.15.4 compliant radio chipset. In our case we used the ED measurement described earlier. Although this measurement is known to be unstable both due to the body motion [14, 7] and the environment changes, we propose to build a rough empirical distance indicator based on smoothed ED measurements, using a series of experiments in dynamic cycling conditions.

In order to assess the channel behaviour between several nodes, we developed a communication platform based on a classical time-slotted protocol, illustrated in Fig.3a which behaves as follows : during the measurement period, each node transmits a packet in a reserved timeslot (according to its identifier), after what it switches to the reception mode in order to collect all the packets sent by the other nodes, and the associated ED measurement. In addition, 2 transmission slots are reserved for control nodes which are used as markers, and for triggering the data storage. This platform self-adapts to the number of nodes, and every node synchronise on the last packet received. Transmission power is set to the maximum value 3 dBm, in order to get the widest measurement range. Thanks to this system, we can obtain the state of all links in the network, in a quasi-simultaneous manner, as the duration of the whole period is 24 ms.

It is important to notice that this platform is only dedicated to preliminary measurements, the platform developed for the implementation of our algorithm is described in 4.2.2.

Using this platform, we made measurements according to the following scenario, as described in Fig.3b: 3 bikes A, B and C, each equipped with one FOX node form a straight line. A is fixed at a distance $d = 0$, while B and C progressively move away from A, until they reach a distance $d = 40m$. The protocol described above is executed during the motion, and every $10m$ a marker is logged on the nodes. This experiment was repeated 10 times, without controlling the cyclists motion. The database obtained consists in about 1000 samples per link at each experimental run.

As expected and illustrated in Fig.4, in spite of a global decrease, the raw ED data collected shows important fast and slow variations and its direct reading would be irrelevant, but considering the relative slowness of a group motion regarding the transmissions, we may be able to obtain a consistent distance information by smoothing the channel measurement over a few seconds. As a rough empirical distance estimation, we propose the combination of a short and long term smoothing on the last ED samples received, based on moving average methods, also taking into account the packet loss. Consider the directed link l_{AB} from A to B and the associated ED measurement signal $s(t_n)$ received at time t_n. Let us first switch $s(t_n)$ to positive values for convenience, say $p(t_n) = s(t_n) - ED_{min}$, as in our case $ED_{min} = -91dBm$. The short term moving average of size W_1, which acts as a high frequency noise remover, is defined as follows : let V_1 be the vector containing at most the W_1 last measurements received within Δ_{t1}:

$$V_1 : \begin{cases} v_i^1 = p(t_{n-i}), & \text{if } (t_n - t_{n-i}) \leq \Delta_{t1} \\ v_i^1 = 0, & \text{else} \end{cases} , 0 \leq i \leq W_1 - 1$$

The short term average $s^1(t_n)$ is:

$$s^1(t_n) = \frac{1}{W_1} \sum_{i=0}^{W_1-1} v_i^1$$

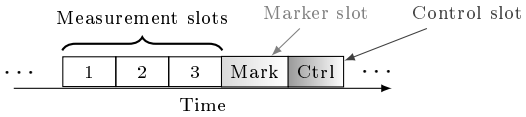
The value of $s^1(t_n)$ is then sampled every Δ_{t2} period and stored in vector V_2, of size $W2$, such that

$$V_2 = [s^1(k\Delta_{t2}), s^1((k-1)\Delta_{t2}), ..., s^1((k - W_2 + 1)\Delta_{t2})]$$

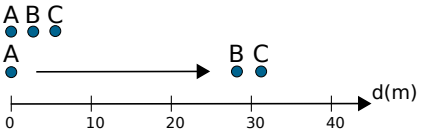
$k\Delta_{t2}$ being the last sampling instant of s^1. Finally the long term average $s^2(t_n)$ is obtained by averaging V_2 :

$$s^2(t_n) = \frac{1}{W_2} \sum_{i=0}^{W_2-1} v_{2,i}$$

This long term average tends to smooth the slower variations (shadowing) due to the motion and environment changes. To limit the quantity of information exchanged in the network, it is finally quantified on 2 bits, according to thresholds th_{20} and th_{30} determined as the values of s^2 at $d = 20m$ and $d = 30m$, averaged between all the 10 realizations. An example of the smoothing and quantization process with the retained parameters is given in Fig.4. Its accuracy is hard to assess, and may be variable, but this method has the advantage to extract a monotonous distance criterion in the case of a group splitting. It also responds to a major constraint of embedded systems, which is the small memory capacities, as in this case only $W_1 + W_2$ samples per link are required, and the moving averages are easily implemented using circular buffers.



(a) Protocol used for the measurement campaign



(b) Links measurement with dynamic distance evolution.

Figure 3: Dynamic distance evolution scenario: Nodes B and C progressively move away from A, meanwhile all the links in the network are being sequentially measured. A marker packet triggered by a specific node is logged each time node B gets past 10 meters.

4.2.2 Algorithm and Protocol Calibration

The practical algorithm we implemented is the fully extended MGMC Algorithm (algorithm 3), with an external beacon periodically sent by the LION routers for new epoch propagation. Let $d(v, j)$ be the quantified proximity estimation computed by node v when receiving a packet from node j. The condition for v accepting j as a mate is if the distance between v and j is smaller than 20m. According to the quantification process, we have the maximum value $M = 3$, and $C_{vj} = 1$ if $d(v, j) \geq 3$. In addition to this, in order to observe interconnections among groups, if $d(v, j) \geq 3$, we don't perform the Max operation upon the whole vector B^v, but only on the B^v_j component, i.e. $B^v_j = Max(B^v_j, d(v, j)$. This will not modify the result of group detection, but add information about the strength of the links between groups, and the nodes at which these weaker links are present. We also added in the transmitted packets the vector $B^v(k_{v-1})$, which corresponds to the last local vector of node v before the current epoch, that supposedly converged. This part of the packet is addressed to the sink: when approaching a group, the mobile router can get the convergence information at epoch $K_N - 1$ without interfering with the bicycle network. This is to avoid more complex protocols such as the exchange of control packets (RTS, CTS, ACK), and the election of a transmitter among the bike nodes. As the design was made for $N = 20$ sensors and 2 bits quantification, each vector B^v is composed of $N \times 2 = 40$ bits, i.e. 5 Bytes.

If we now focus on the communication design, we need to ensure the reception of at least 10 packets per node per second for neighbours in a close communication range, to obtain a correct distance estimation. Our algorithm relies on a random organisation protocol, which is justified by the simple fact that in those conditions synchronised protocols may be very difficult to implement, and not easy to adapt to topology changes. In practice, we experimentally fixed the parameter $T_{max} = 70ms$ as this value ensures a globally fair reception rate around 15 packets per second per node for a static experiment, i.e. all nodes on a table, giving a maximum convergence time of 200 ms. To limit collisions, the implemented communication protocol is based on CSMA/CA without acknowledgement, T_{max} being considered as a random backoff, each node freezing the decrease of T_{max} when sensing the channel busy.

The duration T_P was set to 400 ms to ensure the convergence in moving conditions, i.e. taking into account faulty links. To avoid the propagation of erroneous packets in the network, the AT86RF231 radio chipset implements a CRC-16 error detection algorithm, allowing us to drop the erroneous packets. Bike nodes packets are composed of 21 bytes, which corresponds to

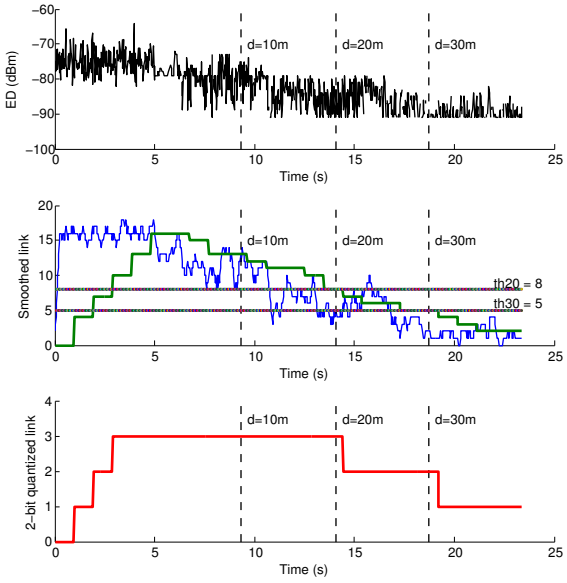


Figure 4: Example of the smoothing and quantification process for link l_{AB} using the retained parameters: $W_1 = 10$, $W_2 = 5$, $\Delta_{t1} = 1s$, $\Delta_{t2} = 1s$, $th_{20} = 8$, $th_{30} = 5$. The black curve is the raw link measurement, blue and green are respectively the short and long term averages, while the red one is the result of quantification.

a transmission time of 656μs.

All the received packets were locally stored on micro-SD cards, as well as additional information, such as the ED value measured for each packet, the local reception timestamp, the number of packets sent every second, and the amount of erroneous (wrong-CRC) packets received.

4.3 Experimental Conditions and Results

We will first explain under which conditions the experiment was realised, before presenting some interesting results extracted from the collected data. The presented results concern the performance of our algorithms for a stable group behaviour, we also show that we were able to dynamically detect group splittings and fusions, as requested for the application.

4.3.1 Experimental Conditions

The experiment was conducted with a group of ten racers, using the global infrastructure represented in Fig.2, in the region of Paris, for about 1 hour, which allowed us to test the reliability of our algorithms and the whole communication platform over time. The circuit was a 2 km loop, in a semi-urban environment, i.e. with both buildings and rather clear areas. As we explained in 4.2.2, for the whole duration of the race every node stored all the packets it received, plus additional data, on its micro-SD card. We could then build an important database of the network, and focus in detail on the behaviour of our algorithms given 2 major racing situations : stable group and dynamic splitting. Indeed, the race started with a long period during which all the racers were forming a unique pack. During this period, one racer shortly moved away before joining in again. After that move, the group split in two sub-groups until the end of the event,

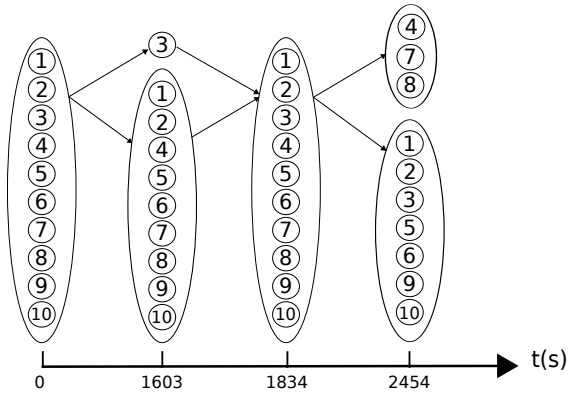


Figure 5: Global progression of the bicycle network during the experiment. We represent here the three main events that were detected and the instants of detection, i.e. isolated racer moving away and joining again the pack, and the pack splitting in two parts.

one motorbike following each formation. The progress of the race was extracted from the stored data, and is described in Fig.5, which validates our platform from the application point of view.

4.3.2 Stability and Performance

We will focus here on the first part of the experiment, for $t \in [0; 1603s]$, during which all the cyclists are riding together, without controlling more their motion, i.e. relative positions and distances may vary. This first step is important to estimate the performances on the full graph before focusing on group splitting. We should first have an estimation on the amount of packets lost during that period. Given the number of packets emitted and the received packets stored by each node, we obtained an average packet loss over all this period of 22%, which is non negligible but seems reasonable given the important traffic and the transmission conditions (motion, bikers acting as obstacles, channel instability...). It is now interesting to focus on the density of the number of messages exchanged before reaching the consensus, and compare it to our theoretical bounds. According to our measurements, during this period the application-oriented graph diameter (i.e. taking into account the distance criterion) is low, $1 \leq \Delta_C \leq 2$, which means that all nodes are almost direct mates. Fig.6 shows the performance distribution. It is interesting to notice that in spite of a non negligible packet loss, the mean value for our algorithm remains lower than the upper bound on the expectation given by 3.2. The shape of that experimental distribution is also in accordance with the one obtained by simulation (Fig.1b).

4.3.3 Dynamic Splitting

After validating our algorithm in terms of convergence performance, we can now observe with more detail how it combines with the distance estimation in real conditions, i.e. evaluate its behaviour in dynamic scenarios, with evolving topologies. From the application point of view, an interesting dynamic scenario is typically when the pack comes to split in case a breakaway occurs. For this we selected the first event described in Fig.5, when node 3 moves away rapidly from the rest of the racers, and analyzed the distance indicators that were computed according to the method described in 4.2.1. Fig.7 shows the evolution of the links between node 3 and its 5 closer neighbours, during that splitting phase. The first observation is that in practice our distance index remains stable and monotonous for the most direct links (i.e. the last decreasing); in this

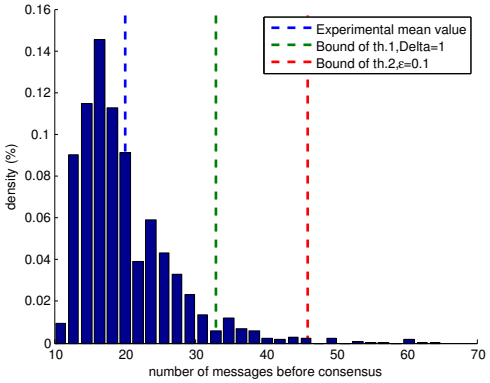


Figure 6: Experimental distribution obtained for the number of messages transmitted before convergence, $N = 10$, $\Delta_C \leq 2$.

case, having the pack forming a single line, every racer behaves as a communication obstacle, which adds uncertainty on the link measurement for nodes located at several hops. The second observation is that after the two groups were detected, a transition phase occurs, during which both groups share the list of their mates, but also the information of an active but weaker link between the groups. After that transition phase, the groups are fully independent.

5 Conclusion

We described in this paper efficient algorithms for group consensus, self-organised and adapted to mobile applications, capable to detect fast topology changes. We provided theoretical bounds on their convergence performances, which were experimentally validated through a challenging application scenario, for which all the functionalities were implemented. One limitation is that in the context of a cycling race, a high proportion of messages are transmitted for the computation of the distance estimation, due to the lack of accuracy of the RSSI measurement. The lack of accurate technological solutions for distance estimation may not be an issue in the future, with the apparition of UWB chipsets implementing time-based distance measurements, which are less dependent to the communication environment. This important network load must be taken into account according to the number of nodes communicating together, to avoid channel saturation and unefficient communications due to a high number of collisions. In the case of a growing number of nodes, it would be necessary to implement mechanisms that control the nodes' communication range, e.g. by adapting the transmission power to the density. As the packet size is also proportional to the number of nodes, the use of adaptive data compression methods could be of interest, in order to reduce the data exchanged without degrading the quality of the measurements.

Acknowledgement

We wish to thank the company Euromedia France for the funding of this project, and for their technical collaboration for the realization of the experiment, giving us the opportunity to design an original experimental scenario with the complete data collection infrastructure. We are also grateful to the company HiKoB, for their important technical support with the design of the platform.

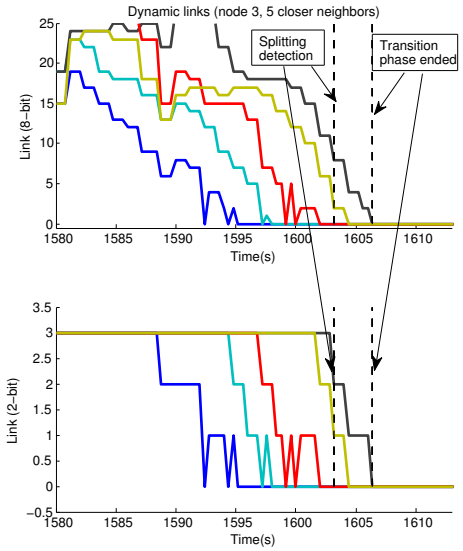


Figure 7: Dynamic link behaviour at node 3 when getting away from the pack (5 closer links). Splitting is detected when all nodes have a weaker proximity index with node 3. During the transition phase, both groups share a weaker link, after what they are out of each other's communication range.

References

[1] "AT86RF231," http://www.atmel.com/devices/at86rf231.aspx, 2012.

[2] "HiKoB FOX sensor," http://www.hikob.com/hikob-fox, 2012.

[3] "HiKoB LION sensor," http://www.hikob.com/hikob-azure-lion, 2012.

[4] I. Amundson and X. Koutsoukos, "A survey on localization for mobile wireless sensor networks," in *Mobile Entity Localization and Tracking in GPS-less Environnments*, ser. Lecture Notes in Computer Science, R. Fuller and X. Koutsoukos, Eds. Springer Berlin Heidelberg, 2009, vol. 5801, pp. 235–254. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04385-7_16[1]

[5] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2748–2761, 2009.

[6] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508–2530, 2006.

[7] N. Chohan and C. Fiorese, "Cyclenet: Empirical analysis of 802.15. 4 in mobile scenarios," 2008.

[8] J. Cortes, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726 – 737, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109807003664

[9] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "Bikenet: A mobile sensing system for cyclist experience mapping," *ACM*

Trans. Sen. Netw., vol. 6, no. 1, pp. 6:1–6:39, Jan. 2010. [Online]. Available: http://doi.acm.org/10.1145/1653760.1653766

[10] F. Iutzeler, P. Ciblat, and J. Jakubowicz, "Analysis of max-consensus algorithms in wireless channels," *Signal Processing, IEEE Transactions on*, vol. 60, no. 11, pp. 6103–6107, 2012.

[11] ——, "Analysis of max-consensus algorithms in wireless channels," *Signal Processing, IEEE Transactions on*, vol. 60, no. 11, pp. 6103–6107, 2012.

[12] S. Kar and J. M. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *Signal Processing, IEEE Transactions on*, vol. 57, no. 1, pp. 355–369, 2009.

[13] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh, "Probabilistic reliable dissemination in large-scale systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 14, no. 3, pp. 248–258, 2003.

[14] M. Lauzier, P. Ferrand, A. Fraboulet, H. Parvery, and J. Gorce, "Full mesh channel measurements on body area networks under walking scenarios," in *Antennas and Propagation (EuCAP), 2013 7th European Conference on*, 2013, pp. 3508–3512.

[15] N. A. Lynch, *Distributed algorithms.* Morgan Kaufmann, 1996.

[16] N. Maréchal, J.-M. Gorce, and J. Pierrot, "Joint estimation and gossip averaging for sensor network applications," *Automatic Control, IEEE Transactions on*, vol. 55, no. 5, pp. 1208–1213, 2010.

[17] R. Motwani and P. Raghavan, *Randomized Algorithms.* New York, NY, USA: Cambridge University Press, 1995.

[18] A. Pal, "Localization algorithms in wireless sensor networks: Current approaches and future challenges," *Network Protocols and Algorithms*, vol. 2, no. 1, pp. 45–73, 2010.

[19] N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 54–69, 2005.

[20] M. Penrose, *Random geometric graphs.* Oxford University Press Oxford, 2003, vol. 5.

[21] S. Čapkun, M. Hamdi, and J.-P. Hubaux, "Gps-free positioning in mobile ad hoc networks," *Cluster Computing*, vol. 5, no. 2, pp. 157–167, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1013933626682

[22] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, ser. IPSN '05. Piscataway, NJ, USA: IEEE Press, 2005.

.1 Proof of theorem 3.1

Before reaching the convergence state, the network will go through a succession of stage S_i, we denote t_i their associate dates (i.e. number of clock cycle). t_i is a random variable representing the date at which the stage goes from S_{i-1} to S_i.

- S_0 is the *starting stage* ($t_0 = 0$).

- S_1 stage is an stage where each node has (at least) received the vector *of the nodes which are at distance 1 with him*, t_1 is the first date at which this situation occurs.

...

- S_i stage is reached when each node has (at least) received the vector of the nodes which are at distance i with him.

It is clear that S_{Δ_G} is the end of the algorithm as every node has received the vector of all other nodes.

Stage S_1 is reached when each nodes has received the B^v vector of its neighbors v, hence it occurs when *all* the nodes have sent their packet at least once as no communication are lost. The well known *coupon theorem* [17] will help us in getting the expectation of t_i. The coupon collector problem indicates, given a random variable taking N possible values with a uniform distribution, the expectation of the number of trial necessary to obtain *each of the* N *values*. This expectation is proved to be Nh_N with $h_n = (1 + \frac{1}{2} + \frac{1}{3} + \ldots + + \frac{1}{N})$. As the event S_1 occurs exactly when *each node* has sent his packet once and as it can be considered that the node that sends his packet is chosen with a uniform distribution, we are exactly faced with a coupon collector problem, and the expectation of t_1 is:

$$\mathbb{E}(t_1) = Nh_N = N(1 + \frac{1}{2} + \frac{1}{3} + \ldots + + \frac{1}{N})$$

Again, being in stage S_1, the transition to S_2 occurs when each node has again sent its B vector, then all node will receive a vector from their neighbor containing the information of the neighbors of their neighbors. Again, we have to wait an additional number of clock cycle $t_2 - t_1$ which expectation is

$$\mathbb{E}(t_2 - t_1 | S_1) = Nh_N$$

.

As we have:

$$\tau = t_{\Delta_G} = t_{\Delta_G} - t_{\Delta_G - 1} + t_{\Delta_G - 1} - t_{\Delta_G - 2} + \ldots t_1 - t_0$$

We can write:

$$\mathbb{E}(\tau) \leq \sum_{i=0}^{\Delta_G - 1} \mathbb{E}(t_{i+1} - t_i | S_i) = \sum_{i=0}^{\Delta_G - 1} Nh_N = N\Delta_G h_N$$

By using the inequality $h_N \leq 1 + log(n)$, we get the result of theorem 3.1:

$$\mathbb{E}[\tau] < N\Delta_G(1 + log(N))$$

.2 Proof of theorem 3.2

Let $A_k^v(t)$ be the event that, during stage S_k, the node v has not send its packet after t iteration. The probability of this event is $\mathbb{P}(A_k^v(t)) = \left(\frac{N-1}{N}\right)^t$: node v did not send his packet during the t time step following the beginning of S_k. If we have $t_{k+1} - t_k \geq t$, it implies that the event $A_k^v(t)$ has taken place for at least one node v, hence using union bound:

$$\mathbb{P}(t_{k+1} - t_k \geq t) \leq \mathbb{P}\left(\bigcup_{v \in G} A_k^v(t)\right) \leq \sum_{v \in G} \mathbb{P}(A_k^v(t)) = \sum_{v \in G} (1 - 1/N)^t$$

Using the fact that $1 - y \leq exp(-y)$ for $0 \leq y \leq 1$, we get:

$$\mathbb{P}(t_{k+1} - t_k \geq t) \leq N\ exp(-t/N)$$

For any $\epsilon \geq 0$, if we choose $t_\epsilon = NLog(N) + NLog\left(\frac{\Delta_G}{\epsilon}\right)$ we get:

$$\mathbb{P}\left(t_{k+1} - t_k \geq NLog(N) + NLog\left(\frac{\Delta_G}{\epsilon}\right)\right) \leq$$

$$N * exp\left(-\left(NLog(N) + NLog\left(\frac{\Delta_G}{\epsilon}\right)\right)/N\right) =$$

$$N * exp(-log(N) - Log(\frac{\Delta_G}{\epsilon})) = \frac{\epsilon}{\Delta_G}$$

Hence, using union bound again, the probability that at least one stage last more than t_ϵ is such that:

$$\mathbb{P}\left(\bigcup_{k=0}^{\Delta_G - 1}(t_{k+1} - t_k \leq t_\epsilon)\right) \leq \sum_{k=0}^{\Delta_G - 1}\mathbb{P}(t_{k+1} - t_k \leq t_\epsilon) \leq \Delta_G.\frac{\epsilon}{\Delta_G} = \epsilon$$

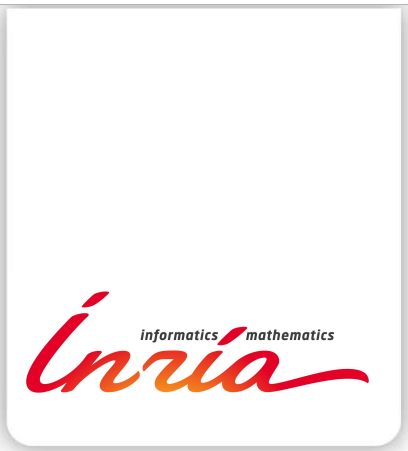
If each stage last less than t_ϵ, it implies that the algorithm will converge in less than $\Delta_G.t_\epsilon$, we have the result:

$$\mathbb{P}(t_{\Delta_G} < \Delta_G.t_\epsilon) \geq \mathbb{P}\left(\bigcap_{k=0}^{\Delta_G - 1}(t_{k+1} - t_k < t_\epsilon)\right) =$$

$$1 - \mathbb{P}\left(\bigcup_{k=0}^{\Delta_G - 1}(t_{k+1} - t_k \geq t_\epsilon)\right) \geq 1 - \epsilon$$

which completes the proof:

$$\mathbb{P}\left(\tau < N.\Delta_G.\left(Log(N) + Log\left(\frac{\Delta_G}{\epsilon}\right)\right)\right) \geq 1 - \epsilon$$



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399