ABSTRACT
This paper deals with distributed algorithms for monitoring the topology of a dynamic group of mobile wireless sensor networks. We propose two major extensions of a distributed static group consensus algorithm and an experimental implementation. Group consensus algorithms are exploited to let each node obtain the knowledge of its connected components. The proposed extensions provide a more accurate information about the proximity of nodes and allow to deal with dynamic 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 real bicycle race. The real traces thus obtained and analyzed show the effectiveness of our live group detection implementation.

Keywords
mobile wireless sensor networks; distributed consensus; dynamic topology estimation

1. INTRODUCTION
Distributed decisions within any group of agents, is a very active research area and theoretical results as well as efficient algorithms have already been proposed [12, 6, 4]. 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 [5, 11].

In this paper, we consider a group of mobile agents moving roughly in a common direction. We propose algorithmic solutions allowing each agent to periodically discover its neighbours: one-hop neighbours as well as multi-hop neighbours. The reference scenario is a bike race, during which groups are susceptible to split or merge. The objective is a live gathering of information about who is present in a group for live TV broadcasting. For that, we need a fully distributed approach allowing every agent to discover with a consensus algorithm the list of neighbours participating to the same pack. This study may be of interest for various other applications such as group navigation support in crowded environments, autonomous navigation of a fleet of robots...

This problem exhibits some similarities with a clustering problem. However, a clustering problem aims at exploiting the structure of a graph and to form some subgroups to ensure a good structure of the network for further communications while our objective is rather to estimate the groups naturally formed in the real world. Hence, we have focused on distributed decision algorithms, which are widely present in the literature. In this context, gossip approaches are very appealing [6, 12, 4]. While some works focused on scaling issues to ensure a proper behaviour when the size of the network grows [9], other works focused on the consensus accuracy and convergence speed [12, 6]. Max-consensus problem has been much less studied than average consensus. In [6], the max-consensus is mentioned as one of the possible consensus operations, but the paper doesn’t provide specific results about the convergence rate. The most relevant contribution is provided by Iutzeler et al. in [7].

The proposed algorithms are based on the N-dimension generalization of the Random Broadcast Max-Consensus algorithm given in [7], allowing each agent to build and share the list of its multi-hop neighbors. We extend this approach to a dynamic context where the group information needs to be updated according to possible group merge or split.

Experimental validation has been done in the context of a cycling race with 10 agents, equipping each bicycle with a wireless sensor node to assess the interactions between the racers and to provide a live monitoring of the dynamic evolution of the cyclists groups that form during the race. We were also able to store the data on the nodes to obtain an accurate database on the network behavior.

The rest of this paper is organised as follows. We present
our mobile group consensus algorithm in section 2. This experiment and the results are presented in section 3 where we provide practical evidence of the efficiency of our Algorithm.

2. CONSENSUS ALGORITHMS FOR MOBILE GROUPS

In this section, we present the mobile group consensus algorithms designed for the cyclist race. Their implementation in a real experimentation are provided in (section 4).

We first present the basic algorithm dealing with a binary notion of neighborhood and with static nodes, which means that the connectivity between nodes does not change over time. We then introduce the mate concept of “close” neighbor (section 2.2) and finally present a solution to manage dynamic networks (section 2.3).

2.1 Static Group Consensus Algorithm

Our first algorithm is a straight extension of a Max-Consensus Algorithm presented in [7]. Iutzeler et al. present a Max-Consensus Algorithm for non-mobile nodes in which each node should become aware of all the nodes. We extend this algorithm so that each node should become aware of all the values held by all other nodes.

The static wireless sensor network is modeled as an undirected graph \( G = (V, E) \). \( V \) is the set of sensors \( |V| = N \) and \( E \) is the set of sensor connections. We denote \( \Delta_G \) the diameter of graph \( G \). In this section, each link is supposed to be error-free and constant in time without collision.

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, \( B^v \) is equal to 1 if nodes \( i \) and \( v \) are in the same connected component.

The sgc algorithm is presented in Algorithm 1. Max is the component-wise maximum operation and we give below an estimation of the convergence time.

**Algorithm 1. Static Group Consensus Algorithm**

*(Executed on each node \( v \))*

1. Initialize \( B^v \) with \( a 1 \) at component \( v \) and \( 0 \) elsewhere
2. repeat
   1. \( T_{\text{max}} = \text{random}(0 - T) \)
   2. While not expired backoff \( T_{\text{max}} \)
      1. receive \((B^j)\) from node \( j \)
      2. \( B^v = \text{Max}(B^v, B^j) \)
   end While
3. broadcast \((B^v)\)
   Until finished

The sgc algorithm convergence rate has been studied in [10]. It is defined as the number of rounds \( \tau \) needed to reach the state where all \( B^v \) are equal to 1, and is such that

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

It is also shown in [10] that an upper bound on \( \tau \) can be given with an arbitrary probability: \(1 - \epsilon\)

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

It is true noting that these theoretical bounds assume a pure random selection at each turn with not any priority, while the implementation exploits a random backoff mechanism which may increase the fairness.

2.2 Group Mate Consensus

Let’s now consider that a node may not take into account the information it receives from a neighbor if a condition between the transmitter and the receiver is not satisfied (e.g. neighbor further than a threshold distance, sporadic connection...) as if the two nodes were not connected. We thus define two nodes as being mates if the application condition is satisfied or if the two nodes have common mates.

The artificial separation of the graph using a proximity criterion (RSSI-based in practice) between the nodes is included in the full algorithm (Algorithm 2). The \( B^v \) vectors have now integer values (rather than boolean): \( B^v \in \{0; M\}, M \in \mathbb{N} \), which represent how close nodes are with respect to the chosen criterion. When the \( v \) node receives a message from node \( j \), it updates its \( B^v \) vector only if \( v \) and \( j \) are mates. The set of edges \( E_C \) associated to the new adjacency matrix \( C \) models the partition of the graph. We also define \( \Delta_G \) the diameter of the graph given by the adjacency matrix \( C \).

2.3 Extension for Node Mobility

When the nodes move, a connected component may break down in two components, or inversely, two independent components may merge. Both are referred to as merge-and-split variations. The mobile network can be modelled as a dynamic graph \( G(t) = (V, E_C(t)) \), but the timescale of these variations is assumed greater than the algorithm convergence time. 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. But, on the contrary, if a node disappears from \( C \), the Max operation cannot propagate this withdrawal. Dynamically forgetting a node would require to share more complex information, thus we rather introduce a periodical reset of all vectors \( B^v \).

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 broadcasting a new epoch indicator: in our case an external source periodically broadcasts an incremented epoch \( K+1 \), which is multi-hop propagated over the network using a similar max-consensus method among the nodes. The Mobile Group Mate Consensus Algorithm (mGMC) is described in Algorithm 2.

**Algorithm 2. Mobile Group Mate Consensus**

*(Executed on each node \( v \))*

1. Initialize \( B^v \) with \( M \) at component \( v \) and \( 0 \) elsewhere
2. Initialize \( k_v \) to 0
3. repeat infinitely
   1. \( T_{\text{max}} = \text{random}(0 - T) \)
   2. While not expired backoff \( T_{\text{max}} \)
      1. receive packet \( P \)
      2. If \( P = \{k_j, B^j\} \) from node \( j \)
Bicycle Network

in details both the data exploitation is performed. In this section we describe to a central sink located on a truck, where the Bicycle Network

surrounding the race aims at collecting the data shared by Local Collection Network

bicycles; the wireless sensor network formed by the nodes located on the composed of three levels. The Bicycle Network

3.1 Experimental setup

inferior to 1 Hz.

count several mobile sinks which may appear or disappear within a few seconds. In addition, we have to take into ac-

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.

A group is considered to split when the distance between

end If

If \( k_j > k_v \) \ //change epoch \( k_v = k_j \)

end If

end If

end While

broadcast \( \{k_v, B^v\} \)
Figure 2: Example of the smoothing and quantification process for link $l_{AB}$ using the retained parameters: $W_1 = 10$, $W_2 = 5$, $\Delta t_1 = 1s$, $\Delta t_2 = 1s$, $th_{20} = 8$, $th_{30} = 5$. 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.

channel busy.

The period duration was set to 400 ms to ensure the convergence in moving conditions, i.e. taking into account faulty links. 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.

3.3 Experimental Conditions and Results

We first explain the experimental setup before presenting some interesting results extracted from the collected data. The performance of our algorithms are presented for a stable group but also in a situation where merge-and-split variations occur, as requested for the application.

3.3.1 Experimental Conditions

The experiment was conducted with a group of ten racers, using the global infrastructure represented in Fig.1, 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 2km loop, in a semi-urban environment, i.e. with both buildings and rather clear areas. As we explained in section 3.2, for the whole duration of the race every node stored all the packets it received, plus additional data, on its micro-SD card. These data provided an important database from which the performance of our algorithms was studied. We focus here only on two major racing situations, the first corresponding to a stable group and the second to a 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 again in. After that move, the group split in two sub-groups until the end of the event, one motorbike following each formation. The progress of the race was extracted from the stored data, and is described in Fig.3 which validates our platform from the application point of view.

3.3.2 Stability and Performance

We will focus here on the first part of the experiment, for $\tau \in [0; 1603 s]$, 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. First of all, 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 number of messages exchanged before reaching consensus, and compare it to the theoretical bounds. According to our measurements, during this period the graph diameter (taking into account the distance criterion) is low, $1 \leq \Delta C \leq 2$, which means that all nodes are almost direct mates. We study the behavior of $\tau$ introduced in section 3.1, which is the number of messages exchanged before algorithm convergence. Fig.4 shows the experimental distribution of $\tau$, which validates the theoretical bounds given in equations (1) and (2) with $\Delta = 1$ and $\epsilon = 0.1$.

3.3.3 Dynamic Splitting

The convergence of our algorithm being validated for a static graph, we now observe with more details how it performs on a dynamic topology by exploiting the distance estimation in real conditions. From the application point of
view, an interesting dynamic scenario is typically when the pack comes to split in case a breakaway occurs. We focus on the first event described in Fig. 4 when node 3 moves away rapidly from the rest of the racers, and we analyze the distance indicators that were computed according to the method described in 3.2.1. Fig. 3 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 despite the fact that, 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 we can observe a short transition phase, during which two groups are detected but with a weak link between them, before being considered fully disjoint. This expected behavior tends to validate the implemented methods in a real life uncontrolled scenario.

4. 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 in 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 inefficient 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 funding 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.

5. REFERENCES

[1] AT86RF231. www.atmel.com/devices/at86rf231.aspx 2012.
[2] HiKoB FOX sensor. www.hikob.com/hikob-fox 2012.
[3] HiKoB LION sensor. www.hikob.com/hikob-azure-lion 2012.
[4] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione. Broadcast gossip algorithms for consensus. IEEE Transactions on Signal Processing, 57(7):2748–2761, 2009.
[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. IEEE Transactions on Information Theory, 52(6):2508–2530, 2006.
[6] J. Cortes. Distributed algorithms for reaching consensus on general functions. Automatica, 44(3):726 – 737, 2008.
[7] F. Iutzeler, P. Ciblat, and J. Jakubowicz. Analysis of max-consensus algorithms in wireless channels. IEEE Transactions on Signal Processing, 60(11):6103–6107, 2012.
[8] S. Kar and J. M. Moura. Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise. IEEE Transactions on Signal Processing, 57(1):355–369, 2009.
[9] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probabilistic reliable dissemination in large-scale systems. IEEE Transactions on Parallel and Distributed Systems, 14(3):248–258, 2003.
[10] M. Lauzier, A. Fraboulet, J. Gorce, and T. Risset. Distributed group consensus algorithms for mobile wireless sensor networks. Technical Report RR-8518, INRIA, 2014.
[11] N. Maréchal, J.-M. Gorce, and J. Pierrot. Joint estimation and gossip averaging for sensor network applications. IEEE Transactions on Automatic Control, 55(5):1208–1213, 2010.
[12] 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, IPSN ’05, Piscataway, NJ, USA, 2005. IEEE Press.