# Split Distributed Computing in Wireless Sensor Networks

*Martin KENYERES [1], Jozef KENYERES [2], Vladislav SKORPIL [1]*

[1] Dept. of Telecommunications, Brno University of Technology, Technická 12, 612 00 Brno, Czech Republic
[2] Zelisko GmbH, Beethovengasse 43-45, A-2340 Mödling, Austria

kenyeres@phd.feec.vutbr.cz, Jozef.Kenyeres@knorr-bremse.com, skorpil@feec.vutbr.cz

**Abstract.** *We have designed a novel method intended to improve the performance of distributed computing in wireless sensor networks. Our proposed method is designed to rapidly increase the speed of distributed computing and decrease the number of the messages required for a network to achieve the desired result. In our analysis, we chose Average consensus algorithm. In this case, the desired result is that every node achieves the average value calculated from all the initial values in the reduced number of iterations. Our method is based on the idea that a fragmentation of a network into small geographical structures which execute the distributed calculations in parallel significantly affects the performance.*

## Keywords

Wireless sensor networks, distributed computing, improvement of distributed computing

## 1. Introduction

The main goal of this paper is to present a novel method which improves the efficiency of distributed computing in wireless sensor networks (WSN). WSN are defined as networks consisting of spatially distributed devices monitoring an environmental quantity [1], [2]. In general, the devices are labeled as nodes. The nodes are assumed to be limited in terms of their computing capabilities and available energy. We implemented the Average consensus algorithm, the distributed algorithm intended to calculate the average value from the initial ones. Because of its simplicity and an iterative manner, this algorithm is suitable for the implementation in WSN, as shown in [3], [4]. On the other hand, the performance of the algorithm may vary [3]. In this paper, we have shown how significantly our proposed method improves the performance of the algorithm. It is based on the idea that a fragmentation of a network into small geographical structures which perform parallel distributed computing significantly affects the performance. The longer computation is executed, the more time nodes have to be in the active mode, which causes higher energy consumption and that they have to either send or receive more messages. This results in the significant energy requirements. The authors of [5] described the impacts of battery exhaustion on WSN and since our novel method saves both time required to execute distributed calculations and the number of transmitted messages, it may be used in order to decrease energy consumption and increase network's lifetime (as described in [6]). As the distribution of the inner states of nodes separated from each other for a longer distance lasts for a long time, before executing the numerical experiments, we expected our novel method to significantly decrease both the number of iterations necessary for the whole network to converge (i.e. necessary time to converge) and the overall number of messages in a network.

In the first part of this paper, the features of distributed computing in WSN and the method of splitting a network into packs have been introduced. In the next chapter, the terms 'the local' and the 'global consensus' have been explained and the proposed mathematical tools to describe our novel method have been introduced. In the numerical experimental part, the performed numerical experiments and analyzed the obtained results have been described.

## 2. Distributed Computing in WSN

In order to describe the properties of WSN, we use a set of graph theory tools [7–10]. We assume that WSN is an undirected graph defined as follows:

$$NET = (\mathbf{V}, \mathbf{E}). \tag{1}$$

*NET* is a label of a network. Each vertex $v_i$ represents a particular node [11] according to the identity number varying from 1 to $N$. $\mathbf{V}$ represents the set of all vertices, i.e. $\mathbf{V} = \{v_1, v_2, \ldots, v_N\}$. Here, the index $N$ also defines the number of vertices; therefore, the size of a network. Some vertices are connected to each other and this connection is referred to as a path (the path is labeled as $e_{i,j}$ in the case of vertices $v_i$ and $v_j$). $\mathbf{E}$ forms a subset of the Cartesian multiplication $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$. Since we use an indirect graph to describe networks, the following statement is valid:

$$e_{i,,j} \in \mathbf{E} \Leftrightarrow e_{j,i} \in \mathbf{E}. \tag{2}$$

As mentioned above, we chose Average consensus as the subject on which we demonstrated our proposed

method. Average consensus is a distributed algorithm calculating the average from a set of initial values [12]. This means that every node forming the network converges to $x_i(k_l)$ defined as follows:

$$x_i(k_l) = \bar{x} = \frac{1}{N}.\sum_{j=1}^{N} x_j(k) \text{ for } k = 1,\ i = 1,2,...,N. \quad (3)$$

Here, $i$ and $j$ represents the indices of corresponding vertices, i.e. $v_i \in \mathbf{V}$ and $v_j \in \mathbf{V}$. The value of $k_l$ represents the last iteration in which a network achieves the consensus. The vector $\mathbf{x} \in R^N$ contains the inner values of nodes and is updated for each $k$.

The consensus is reached in an iterative, distributed manner [12]. This means that every node converges to the average according to [13]:

$$x_i(k) = x_i(k-1) + \varepsilon \cdot \sum_{j=1}^{N} \{[x_j(k-1) - x_i(k-1)] \cdot A_{ij}\}. \quad (4)$$

Here, $\mathbf{A} \in \{0, 1\}^{N \times N}$ is the adjacent matrix [14],[15]. If $v_i$ and $v_j$ are neighbors, then $A_{ij}$, $A_{ji} = 1$, otherwise, they are equal to 0. Only inner value of a node and values sent by adjacent nodes are locally available. The value of $\varepsilon$ determines the speed of the convergence of the algorithm [16]. The convergence range is determined as follows:

$$0 < \varepsilon \leq \frac{1}{\max\{w\}}. \quad (5)$$

Here $w_i$ is the weight of a node and represents the number of $i$'s adjacent nodes. For the node represented by the vertex $v_i$, it is defined as follows:

$$w_i = \mathbf{A}_{ij} \times \mathbf{J}_{1,N}^T. \quad (6)$$

Here, $\mathbf{J}$ is an all-ones matrix [17].The parameter $w_i$ determines the number of the nodes adjacent to node $i$.

Mathematically, the convergence can be explained as follows [18]:

$$\lim_{k \to \infty} x_i(k) = \frac{1}{N} \cdot \sum_{j=1}^{N} x_j(1). \quad (7)$$

It is not possible to fulfill this condition; therefore, we have defined an event when a node is considered to be converged. We used the distributed mechanism described in [3], which allows every node to determine the event of convergence (i.e. consensus) in a distributed manner. After it reaches the consensus, it no longer upgrades its inner value.

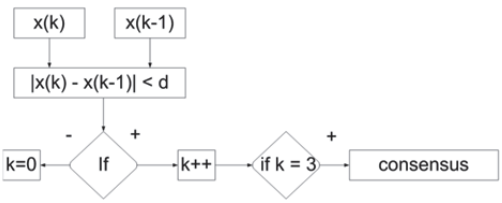


**Fig. 1.** Condition of distributed classification of reaching consensus.

We defined the parameter $d$, whose value determines the precision of the algorithm. A node compares its current inner value with the one from the previous iteration. When their difference is smaller than $d$ during the three following iterations, the node considers itself to be converged.

## 2.1 Split Distributed Computing

Split distributed computing is based on the idea that nodes situated in a geographically close area are gathered into a pack. The pack is an entity consisting of geographically proximate nodes. In case of the Average consensus, each pack converges to the local average determined by the initial values of the nodes forming this pack. Every pack appoints a head, i.e. a node of the highest connectivity within the pack. In this paper, a node's connectivity is defined according to the weight $w_i$.

The node in a pack fulfilling this condition is appointed as the head:

$$\exists v_i \in \mathbf{PK_x}, \nexists v_j \in \mathbf{PK_x} : w_j > w_i \Rightarrow H_i = 1. \quad (8)$$

*PK* defines the label of a pack. **H** is the vector which contains 1 (when $v_i$ is a head) or 0 (when it is not). This definition implies that the node whose weight is the highest within a pack is appointed as the head. When there is not another node (labeled as $v_j$) in the pack $\mathbf{PK_x}$ with weight larger than $v_i$'s, it results in choosing $v_i$ for the head of the pack (therefore, the appertaining position within **H** matrix is set to 1).

After the local consensus is reached within each pack, the phase of reaching the global consensus begins and just the heads of particular packs communicate together and converge to the average. Each node has to fulfill this condition:

$$\forall v_i \in \mathbf{V}, \exists \mathbf{PK}_x, \nexists \mathbf{PK}_y : v_i \in \mathbf{PK_x} \wedge v_i \in \mathbf{PK}_y \text{ for } x \neq y. \quad (9)$$

This means that every node is included only in one pack. The definition implies that every node affects the average value of exactly one pack. If it were present in more than one pack, the final result would differ from the expected results defined by (3). Thus, splitting a network into the packs has to fulfill these conditions:

$$\bigcup_{x=1}^{Q} \mathbf{PK_x} = \mathbf{V} \wedge \bigcap_{x=1}^{Q} \mathbf{PK_x} = \mathbf{\emptyset}. \quad (10)$$

This means that the packs contain all nodes and there is no node which is shared by any other packs. It implies that every node is assigned to just one pack. If it were present in either more than one or no one, the incorrect result would be achieved. Only nodes forming a same pack communicate with one another in the phase of reaching the local consensus. The messages from nodes forming other packs are rejected and do not affect the pack's inner state. The union of all the packs forms the whole set **V**. Their conjunction results in an empty set because no pack shares at least one node with any other pack.

### 2.2 Distributed Reaching the Local Consensus

In this part, we focus on the reaching the local consensus. This phase is executed independently and in parallel in each pack. We assume that nodes acquaint just adjacent nodes forming the same pack with their inner value. Just nodes fulfilling the condition (11) communicate with the node labeled as $v_i$:

$$\mathbf{NP_i} = \{v_i \in \mathbf{V} \big| (v_i, v_j), (v_j, v_i) \in \mathbf{E} \wedge p_{ij}, p_{ji} = 1\}. \quad (11)$$

Here, **p** is the matrix determining whether two nodes are in the same pack (if they are, $p_{ij}$, $p_{ji} = 1$) and **NP** is a set containing all nodes with which $v_i$ communicates during the phase of reaching the local consensus.

Each node converges iteratively to the local average determined as follows:

$$y_{lci} = \{\sum_{j=1}^{N} [x_j(1) \cdot p_{ij}] + x_i(1)\} \cdot \{\mathbf{p_{ij}} \times \mathbf{J}_{\mathbf{1,N}}^{\mathbf{T}} + 1\}^{-1}. \quad (12)$$

The value closed to (12) is achieved in a distributed manner as follows:

$$x_i(k) = x_i(k-1) + \varepsilon \cdot \sum_{j=1}^{N} \{[x_j(k-1) - x_i(k-1)] \cdot A_{ij} \cdot p_{ij}\}. \quad (13)$$

Let $\mathbf{K} \in N^Q$ be the vector containing all the values of $k$, i.e. the number of the iteration, in which a single pack reaches a local consensus:

$$\mathbf{K} = \{k_1, k_2, ....k_Q\}. \quad (14)$$

The parameter $Q$ is the number of packs in the network:

$$Q = \mathbf{H} \times \mathbf{J}_{\mathbf{1,N}}^{\mathbf{T}}. \quad (15)$$

After all packs reach the local consensus, the network as the whole converges to global consensus. This moment of transmission is defined as follows:

$$\exists k_x \in \mathbf{K}, \nexists k_y \in \mathbf{K} : k_y > k_x. \quad (16)$$

We label $k_x$ fulfilling the condition (16) as $k_{lc}$. All packs reach the local consensus in parallel. After all the packs reach the consensus, there are $Q$ local consensuses. Since the phase of reaching the global consensus can begin after all the packs converge, the local consensus which lasted for the longest time determines the length of this phase. We insert all the durations within one process into the set of **K**. The maximal value within this set determines $k_{lc}$ and therefore, the duration of the whole phase of reaching the local consensuses. It determines the number of iterations necessary for the slowest pack in a network to reach the consensus. In this moment, all packs reached their local averages.

### 2.3 Distributed Reaching the Global Consensus

In this phase, just the heads continue to communicate and send messages to each other. The initial value of each head is determined by the particular local consensus. Each head converges to the average calculated from all the local consensuses:

$$y = \frac{\sum_{i=1}^{N} y_{lci}}{Q} = \sum_{i=1}^{N} \{\sum_{j=1}^{N} [x_j(1) \cdot p_{ij}] + x_i(1)\} \cdot [(\mathbf{p_{ij}} \times \mathbf{J}_{\mathbf{1,N}}^{\mathbf{T}} + 1) \cdot Q]^{-1}\}. \quad (17)$$

The heads converge to these values in such a way that they update their inner value in a distributed manner, which can be defined as follows:

$$x_i(k) = H_i \cdot \{x_i(k-1) + \varepsilon \cdot \sum_{j=1}^{N} \{[x_j(k-1) - x_i(k-1)] \cdot P_{ij} \cdot H_j \cdot [\sum_{q=1}^{N} p_{jq} + 1]\}. \quad (18)$$

Here, **P** is a two dimensional matrix and determines whether two packs' heads are adjacent. Mathematically, it can be defined as follows:

$$\begin{aligned} &\exists v_i \in \mathbf{PK_x}, \exists v_j \in \mathbf{PK_x} : A_{ij}, A_{ji} = 1 \Rightarrow \\ &\forall v_t \in \mathbf{PK_x}, \forall v_l \in \mathbf{PK_y} : P_{tl}, P_{lt} = 1 \end{aligned} \quad (19)$$

We assume that transmitting a message to an adjacent head requires a single iteration. After this phase is completed, the whole process is completed and the network reaches the consensus. The previous condition implies that two heads are adjacent only if there is at least one node in both these packs containing the respective adjacent heads which is adjacent to the node in the other pack.

This formula means that if there is $v_i$ from the pack $\mathbf{PK_x}$ and $v_j$ from the pack $\mathbf{PK_y}$ and they are adjacent, the indices within **P** matrix belonging to all the nodes from both these packs are set to 1.

## 3. Numerical Experiments

In this section, our method is verified using the numerical experiments performed in Matlab.

### 3.1 Network with a Tree Topology

In the first numerical experiment, we presented our method on the network whose topology was the tree with the size of 15 nodes and step-by-step explained particular steps. The topology of this network is shown in Fig. 4. We can see that the network is formed by the nodes which have

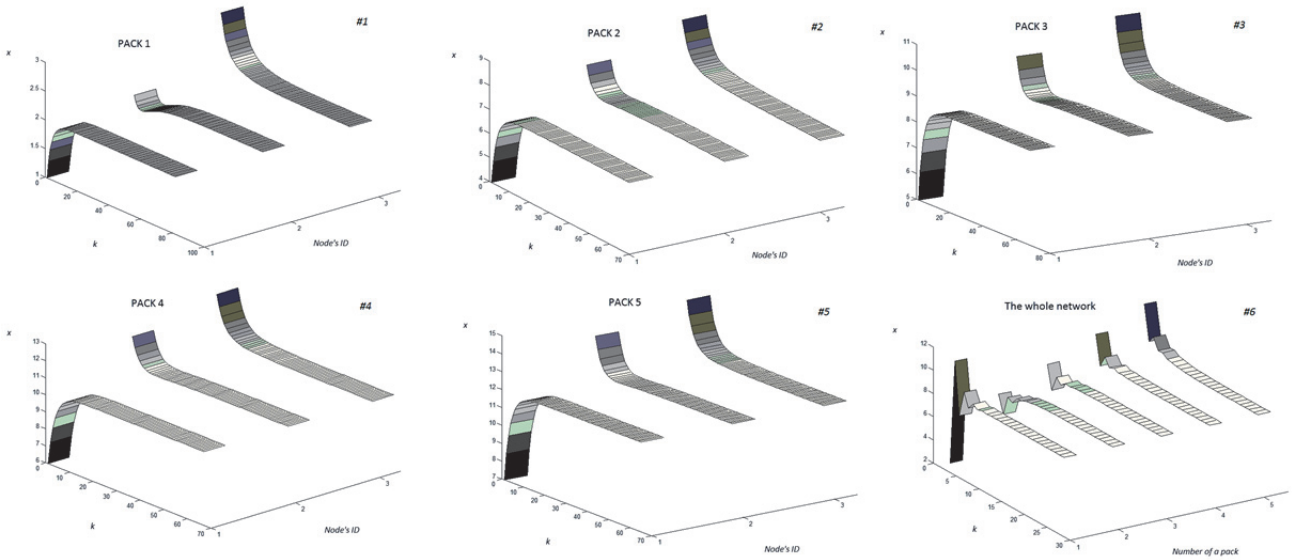


**Fig. 2.** The graphs depicting behavior of inner values within particular packs.

either three or just one neighbor. Nine of them have only one neighbor and the rest of six have three neighbors. We can see that the average number of neighbors equals just 1.8 neighbors per a node. Therefore, this topology is considered to be less connected.

### 3.1.1 Iterations Minimization

As the first step, we divided the network into the small packs whose nodes then converged to the local consensuses. (The local consensuses of these packs differ from each other if the initial values of the nodes forming these packs are not same). Following the pack description presented in Sec. 2, it was also necessary to choose the head of each pack. We see can in Fig. 5 that choosing the head $h$ according to $w$ parameter is clear for $\mathbf{PK}_2$, $\mathbf{PK}_3$, $\mathbf{PK}_4$ and $\mathbf{PK}_5$. A small problem occurs when the nodes try to appoint the head of $\mathbf{PK_1}$ because there are two nodes with maximal $w$. In such a case, the node whose identity number is of the lower value will become the head. This procedure allows us to solve this ambiguity.

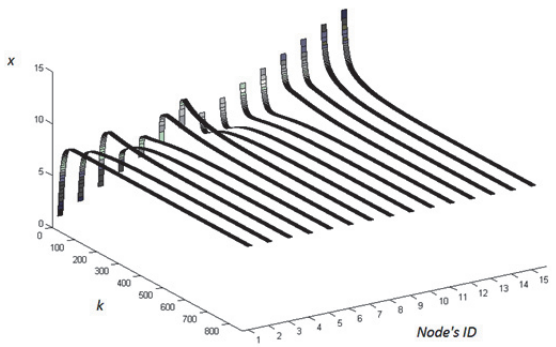


**Fig. 3.** The graphs depicting behavior of inner values when the Standard method was used.

We executed the Average consensus algorithm twice in this network. For the first time, we used the Standard method and the algorithm converged the way described in [3]. The network converges as the aggregate to the average counted from the initial values. In this way, a node is able to reach the average value just according to messages sent by the adjacent nodes and inner state from previous iteration. The values of particular node in every iteration are shown in Fig. 2, parts #1–#5. Communication among the nodes is depicted by solid lines in Fig. 5**.**

In the second case, we used the Partial method, where every **PK** reached the local consensus according to (18).

After this phase is completed, the topology was in fact changed to the one shown in Fig. 5 (communication among the heads is depicted by dash lines) where each **PK** was substituted by the corresponding head $h$ whose initial value was determined by a particular local consensus. The values of particular heads varied during the iterations are shown in Fig. 2, part #6 (each head obtains the value of the local consensus).

We can see from the results shown in Tab. 1 that the pack labeled as $\mathbf{PK_5}$ reached the local consensus as the last. So, the phase of reaching the local consensus ends in 86$^{th}$ iteration. Consequently, the second phase began and re-

| Method | $k_1$ | | | | $sm$ | | |
|---|---|---|---|---|---|---|---|
| Standard | 860 | | | | 12900 | | |
| Partial achieving the consensus | *k of packs* | | $k_{gc}$ | $k_1$ | | | |
| | $\mathbf{PK_1}$ | 83 | 27 | 86 + 27 = 113 | 249 | 297 | 1422 |
| | $\mathbf{PK_2}$ | 71 | | | 213 | | |
| | $\mathbf{PK_3}$ | 69 | | | 207 | | |
| | $\mathbf{PK_4}$ | 66 | | | 198 | | |
| | $\mathbf{PK_5}$ | 86 | | | 258 | | |

**Tab. 1.** Table containing the results for the first numerical experiment.

sulted in the global consensus – the average value counted from the initial values of the nodes. We can see that this phase lasted only for 27 iterations. Then the overall number of the iterations $k_l$ is counted as addition:

$$k_l = k_{lc} + k_{gc}. \tag{20}$$

Here, $k_{gc}$ represents the number of iterations required to achieve the global consensus. The parameter $k_{lc}$ determines the number of iterations required by the phase of reaching the local consensus. Obviously, the results from Tab. 1 show that our method rapidly decreased both the number of iterations and the number of the sent messages. We saved 747 iterations using our proposed method and 11478 messages. In [19–20], the rate of algorithm reaches hundreds of the iterations necessary for a network to reach the consensus (therefore, $k_l$) just like in our experiments (when the Standard method was used). Comparing two networks with significantly different attributes is ambiguous because of the numerous aspects affecting $k_l$ such as the topology of a network, the connectivity, the maximal hop distance etc.). The deep analysis is shown in [21–22].

Messaging is explained in detail in the next section.

### 3.1.2 Analysis of Sent Messages

We assume that nodes send the broadcast messages in order to transmit information about their inner state. Therefore, in the Standard method, the overall number of messages *sm* is determined as follows:

$$sm = N \cdot k_l. \tag{21}$$

In our numerical experiment, *sm* value is calculated as follows:

$$sm = 15 \cdot 860 = 12900. \tag{22}$$

In the Partial method, the way of determining the number of messages differs. In the phase of reaching the local consensus, the node sends messages until its pack achieves the consensus. During the phase of reaching the global consensus, the heads may not be mutually adjacent; therefore, delivering a message could require sending more than one message. When the head of **PK$_1$** (Head 1) wants to inform other heads about its inner state, it sends a broadcast message to all its adjacent nodes (in our case, it is Head 2, Head 3 and Node 1), by which it transmits information to Heads 2 and 3. Subsequently, the message has to be retransmitted by Node 1 to Node 3, which retransmits it again in order to deliver that information to Head 4 and 5. As we assume a broadcast transmission mode, just one message is sufficient for information to be delivered to both Head 4 and 5 from the Node 1. Thus, three messages are necessary for Head 2 to send information about its inner state to all the other heads. This procedure is shown in Fig. 4.

According to the previous description, we calculated the overall number of the sent messages:

$$sm = (3.8 + 3.71 + 3.69 + 3.86) + 27 \cdot (3 + 1 + 1 + 3 + 3) = 1422. \tag{23}$$

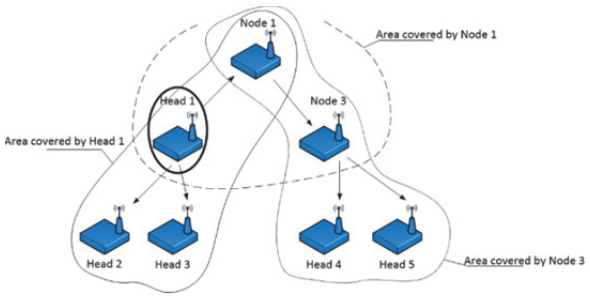


**Fig. 4.** Communication of Head 1 with other heads.

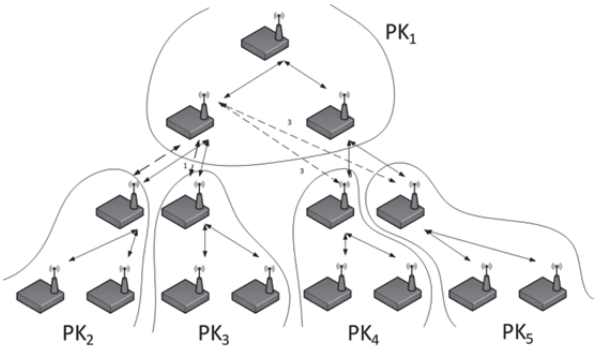


**Fig. 5.** Fragmentation of the network into packs.

## 3.2 Networks with a Random Topology

In the second numerical experiment, we used the generator described in [23] to generate a random topology network. We set its size to 24 nodes and the size of the network's area and the nodes' communication range were set with such value that the network can be classified as an average density network. The network topology is always connected, i.e. every node is able to communicate (single-hop or multi-hop) with any other node. The example is shown in Fig. 6. We present the results achieved from seven numerical experiments. For each numerical experiment, we used the same topology and changed the sizes of packs. The parameter ε was set to 0.08. The network's division into packs is shown in Fig. 7.

The scenario 1 is a case when the algorithm was executed without splitting the network. The network in the scenario 2 is divided into eight packs of the size of three nodes per pack. In the scenario 3, we created six packs consisting of four nodes. In the scenario 4, we used the network consisting of five packs with varying sizes. There are

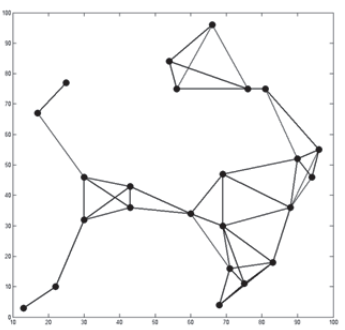

**Fig. 6.** Scenario 1 - the network was not split.

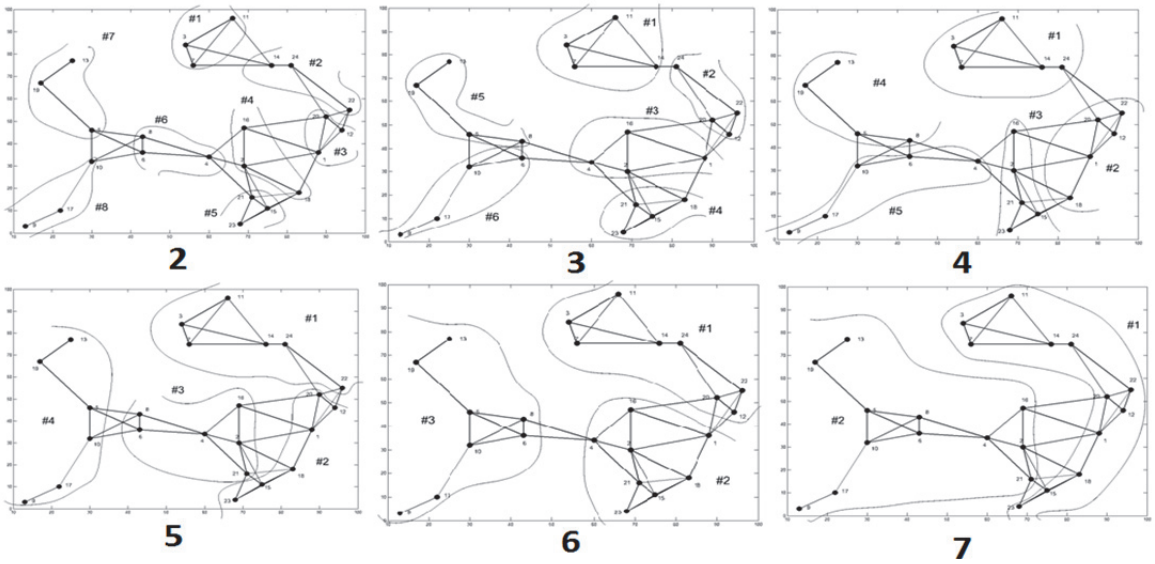


**Fig. 7.** Figure containing each scenario executed in this paper.

four packs with the size of five nodes per pack and one with four nodes. In the scenario 5, the network consisted of four packs whose size was six nodes per pack. The network in the scenario 6 has three packs with the size of eight nodes per a pack. In the last scenario, the network consisted of two packs, both containing twelve nodes.

From the results shown in Tab. 1, it is obvious that our proposed method significantly decreases number of iterations necessary for WSN to converge $k_l$. Next, we can see that in case when the network is divided into more packs, reaching the local consensus requires less iterations. However, the phase of reaching global consensus was much faster for the networks containing a few numerous packs. We can see that splitting distributed computing is effective for a network divided into smaller packs. However, we are not able to claim which scenario is the best because there are a lot of factors affecting $k_l$. The best results were obtained for the scenario 2, where the network needs by 72.09% less iterations compared with the case when the network was not split. It is a significant improvement of distributed computing without negatively affecting the result's precision. In the networks containing bigger packs, we decreased $k_l$ by approximately 50 %. For the network formed by huge packs, the method is less effective. We saved just 15.72% of iterations. We repeated this procedure for other 19 networks whose size and features were both the same and calculated average saved $k_l$ from all 20 networks. We can see from the results shown in Tab. 3 that the method decreases $k_l$ regardless the number of packs, but the higher number the network is divided into, the more effective the method is. We can see that the scenarios in which the network is formed by higher amount of packs are more effective.

| Scenario | Percentage of saved iterations [%] | Percentage of saved messages [%] |
|---|---|---|
| 2 | 71.42 | 83.25 |
| 3 | 66.58 | 81.79 |
| 4 | 54.92 | 74.05 |
| 5 | 50.59 | 73.45 |
| 6 | 43.8 | 72.62 |
| 7 | 13.54 | 38.47 |

**Tab. 2.** Table containing comparison of scenarios.

# 4. Conclusion

In this paper, we introduced a novel method to accelerate distributed computing in WSN and analyzed its performance. The major idea is that a fragmented network consisting of a group of smaller elements performs the distributed computation much more efficiently. Thus, we divided the network into the packs. We changed the number and the size of packs and compared the results obtained using the well-known distributed algorithm. We depicted a deep analysis for one network. Then we repeated same procedures for other 19 networks and calculated the average of saved iterations. We can see from Tab. 2 that the best results were achieved when the network was divided into a larger number of smaller packs. In that case, we achieved 71.42 % reduction of iterations in the average and 83.25 % reduction of the number of the sent messages. With increase of the size of the packs, the reduction was decreasing. We also see other important features of this method: $k_{lc}$ is the smallest for networks formed by lot of packs, but $k_{gc}$ for networks formed by a small number of packs. These results encourage us to try to improve also this aspect.

# Acknowledgments

Research described in this paper was financed by the National Sustainability Program under grant LO1401. For the research, infrastructure of the SIX Centre was used. This work was also supported by the project FEKT-S-14-2352 Research of electronic, communication and information systems.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pack #1 | - | 37 | 27 | 117 | 241 | 351 | 614 |
| Pack #2 | - | 102 | 53 | 25 | 238 | 71 | 330 |
| Pack #3 | - | 38 | 53 | 208 | 107 | 311 | - |
| Pack #4 | - | 91 | 24 | 190 | 356 | - | - |
| Pack #5 | - | 33 | 190 | 297 | - | - | - |
| Pack #6 | - | 35 | 179 | - | - | - | - |
| Pack #7 | - | 108 | - | - | - | - | - |
| Pack #8 | - | 83 | - | - | - | - | - |
| $k_{lc}$ | - | 108 | 190 | 297 | 356 | 351 | 614 |
| $k_{gc}$ | - | 98 | 30 | 49 | 26 | 10 | 8 |
| $k_l$ | 738 | 206 | 220 | 346 | 382 | 361 | 622 |
| Comparison $k_l$ with 1 scenario | 0 | 532 | 518 | 392 | 356 | 377 | 116 |
| Percentage of saved iterations [%] | - | 72.09 | 70.19 | 53.12 | 48.24 | 51.08 | 15.72 |
| Comparison $k_{lc}$ with minimum | - | 0 | 82 | 189 | 248 | 243 | 506 |
| Comparison $k_{gc}$ with minimum | - | 90 | 22 | 41 | 18 | 2 | 0 |
| Number of sent messages | 17712 | 2953 | 2524 | 4436 | 5730 | 5884 | 11344 |

**Tab. 3.** Table containing results for each scenario and pack.

# References


[1] FERRI, R., KIM, M., YEE, E. *Wireless Sensor Network*. U.S. Patent Application 10/856,684.

[2] YANG, S.-H. *Wireless Sensor Networks*. London Springer, 2014. DOI: 10.1007/978-1-4471-5505-8

[3] KENYERES, J., KENYERES, M., RUPP, M., et al. WSN implementation of the average consensus algorithm. In *11th European Wireless Conference 2011-Sustainable Wireless Technologies (European Wireless)*. Vienna (Austria), 2011, p. 1–8. ISBN: 978-3-8007-3343-9

[4] CONTRERAS, R., RESTREPO, S. E., PEZOA, J. E. Implementing the distributed consensus-based estimation of environmental variables in unattended wireless sensor networks. In *SPIE Proceedings,* Vol. 9248. Amsterdam (Netherland), Oct. 2014, p. 321–329. DOI: 10.1117/12.2069822

[5] YUGANDHAR, B., KRISHNAIAH, P. Enhancing the life time of sensor node in a wireless sensor network. *International Journal of Scientific Engineering and Technology Research*, 2014, vol. 3, no. 43, p. 8631–8635. ISSN: 2319-8885

[6] BAKR, B. A., LILIEN, L. T. Extending lifetime of wireless Sensor networks by management of spare nodes. *Procedia Computer Science*, 2014, vol. 34, no. 1, p. 493–498. DOI: 10.1016/j.procs.2014.07.053

[7] BIGGS, N. *Algebraic Graph Theory*. 2nd ed., rev. Cambridge (UK): Cambridge University Press, 1993. DOI: 10.1017/cbo9780511608704

[8] ANDRÁSFAI, B. *Graph Theory: Flows, Matrices*. 1st ed. Boca Raton (FL, USA): CRC Press, 1991. ISBN: 0852742223.

[9] FOULDS, L. R. *Graph Theory Applications*. 1st ed. New York (USA): Springer Verlag, 1992. DOI: 10.1007/978-1-4612-0933-1

[10] BENJAMIN, A., CHARTRAND, G., ZHANG, P. *The Fascinating World of Graph Theory*. Princeton (NJ, USA): Princeton University Press, 2015. ISBN: 9780691163819

[11] SWAMI, A., ZHAO, Q., HONG, Y., et al. *Wireless Sensor Networks: Signal Processing and Communications*. 1st ed. Hoboken (NJ, USA): John Wiley & Sons, 2007. ISBN: 9780470035573

[12] BÉNÉZIT, F. Distributed average consensus for wireless sensor networks. *PhD Thesis*. École Polytechnique Fédérale de Lausanne, Switzerland, 2009.

[13] PRIOLO, A., GASPARRI, A., MONTIJANO, E., et al. A distributed algorithm for average consensus on strongly connected weighted digraphs. *Automatica*, 2014, vol. 50, no. 3, p. 946–951. DOI: 10.1016/j.automatica.2013.12.026

[14] GIBBONS, A. *Algorithmic Graph Theory*. 5th ed., rev. New York (USA): Cambridge University Press, 1985. ISBN: 0521288819

[15] BAPAT, R. B. *Graphs and Matrices.* New York (NY): Springer, 2010. DOI: 10.1007/978-1-84882-981-7

[16] KENYERES, J., KENYERES, M., RUPP, M. Experimental node failure analysis in WSNs. In *IEEE 18th International Conference on Systems, Signals and Image Processing (IWSSIP) 2011.* Sarajevo (BH), 2011, p. 1–5. ISBN: 9781457700743

[17] HORN, R., JOHNSON, CH. *Matrix Analysis*. 2nd ed., rev. New York (NY): Cambridge University Press, 2012. ISBN: 9780521839402

[18] XIAO, L., BOYD, S., KIM, S.-J. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing,* 2007, vol. 67, no. 1, p. 215–233. DOI: 10.1016/j.jpdc.2006.08.010

[19] KENYERES, M., KENYERES, J., ŠKORPIL, V. *Effect of the Speed of the Algorithm's Convergence on the Quality of Distributed Computing in WSN*. 5 pages. [Online] Cited 2015-06-26. Available at: http://access.feld.cvut.cz/view.php?nazevclanku=effect-of-the-speed-of-the-algorithms-convergence-on-the-quality-of-distributed-computing-in-wsn&cisloclanku=2015040001

[20] SLUČIAK, O., HILAIRE, T., RUPP, M. A general formalism for the analysis of distributed algorithms. In *IEEE International Conf. on Acoustics Speech and Signal Processing (ICASSP).* Dallas (TX, USA), 2010, p. 2890–2893. DOI: 10.1109/icassp.2010.5496169

[21] KENYERES, M., KENYERES, J., ŠKORPIL, V. *Effects of System Topologies' Attributes on Average Consensus Algorithm - part I.* 10 pages. [Online] Cited 2015-07-01. Available at: http://access.feld.cvut.cz/view.php?nazevclanku=effects-of-system-topologies%E2%80%99-attributes-on-average-consensus-algorithm-part-i&cisloclanku=2015060002

[22] KENYERES, M., KENYERES, J., ŠKORPIL, V. *Effects of System Topologies' Attributes on Average Consensus Algorithm - part 2.* 7 pages. [Online] Cited 2015-07-01. Available at: http://access.feld.cvut.cz/view.php?nazevclanku=effects-of-system-topologies%E2%80%99-attributes-on-average-consensus-algorithm-part-ii&cisloclanku=2015060003

[23] KENYERES, J., KENYERES, M., RUPP, M., et al. Connectivity-based self-localization in WSNs. *Radioengineering*, 2013, vol. 22, no. 3, p. 818–827.

## About the Authors …

**Martin KENYERES** was born in Bratislava, Slovakia. He received his M.Sc. from the Slovak University of Technology in Bratislava in 2013. His research interests include distributed computing and wireless sensor networks. In 2013, he was with TU Vienna, Austria, where he participated in NFN SISE project under Professor Markus Rupp's supervision. Since 2014, he has been with Brno University of Technology, where he works towards his PhD thesis.

**Jozef KENYERES** was born in Bratislava, Slovakia. He received his Ph.D. from the Slovak University of Technology in Bratislava in 2014. His research interests include embedded systems and wireless sensor networks. From 2006 to 2009, he was with Slovak Telecom, from 2009 to 2013, he worked as a project assistant at TU Vienna and since 2014 he has been with Zelisko GmbH, where he works as a software developer.

**Vladislav ŠKORPIL** was born in Brno, Czech Republic. He attended Brno University of Technology, Faculty of Electrical Engineering, Dept. of Telecommunications. From 1980 to 1982 he worked as a designer for the telecommunication design office. Since 1982, he has been with BUT. His research interests include modern telecommunication systems. He has published more than 110 international scientific papers.