Basic Computations in Wireless Networks*

Ioannis Caragiannis[1], Clemente Galdi[1,2], and Christos Kaklamanis[1]

[1] Research Academic Computer Technology Institute
Department of Computer Engineering and Informatics
University of Patras, 26500, Rio Greece
[2] Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Universitá di Salerno, 84081, Baronissi (SA), Italy

Abstract. In this paper we address the problem of estimating the number of stations in a wireless network. Under the assumption that each station can detect collisions, we show that it is possible to estimate the number stations in the network within a factor 2 from the correct value in time $O(\log n \log\log n)$. We further show that if no station can detect collisions, the same task can be accomplished within a factor of 3 in time $O(\log^2 n)$ and maximum energy $O(\log n)$ per node, with high probability. Finally, we present an algorithm that computes the minimum value held by the stations in the wireless network in time $O(\log^2 n)$.

1 Introduction

In the last years wireless networks have attracted a lot of attention of the scientific community. Such networks essentially constitute large scale dynamic distributed systems in which each node has a low computational power and limited lifetime. An extreme example of wireless networks are sensor networks in which, thousands of low-cost, independent entities are deployed in an area and their task is to self-organize a network in order to accomplish a specific task. Each node is equipped with sensing devices and, depending on the specific task, the sensors may be identical or there may be different kinds of nodes. The communication among the nodes is guaranteed by means of radio or laser transmitters/receivers.

One of the assumptions that facilitate the design of algorithms in wireless networks is the knowledge of the number (or an upper bound on the number) of stations in the network. However, especially in highly dynamic networks this assumption seems to be strong.

Network models. The radio network consists of n *stations*, devices running on batteries and with low computational capabilities. The stations communicate by means of a shared channel. The stations are anonymous, in the sense that they do not have, or it is not feasible to retrieve any ID or serial number.

Each station has a local clock and all the clocks are synchronized. Although such devices may have limited computational capabilities, this is not a strong assumption because of the existence of various protocols for clock synchronization

* This work was partially supported by the European Union under IST FET Integrated Project 015964 AEOLUS.

(e.g., [4]). During each time step, each station can send and/or receive a message. We assume a *reliable single-hop* network, that is, whenever exactly one station transmits a message, all the other stations receive it. If two or more stations send a message in the same time slot, a *collision* occurs. We distinguish three different models, depending on the information obtained by a station whenever a collision occurs: (a) *CD*: All the stations can detect collisions; (b) *strong-noCD*: The stations that send a message can detect a collision while the other stations cannot distinguish between channel noise and a collision; (c) *weak-noCD*: No station can detect a collision.

As usual, the time needed by an algorithm to execute a task is one performance measure we will consider. Furthermore, since stations have limited lifetime, energy saving is an important issue to consider. We assume that each station can be in two different states: *active/awake* or *inactive/sleep*. When a station is active, it can send and or receive messages and execute computations and we assume consumes energy 1 for each time step. When a station is in a sleep state it neither sends/receives messages nor executes computations. A station, before switching to the sleep state, sets a timer. Whenever the timer expires, the station goes back to the active state. In the sleep state we assume that the station consumes no energy.

Previous work. Various computation problems have been studied in the literature in the above models. In the case of reliable radio networks, specific solutions have been designed for computing the function sum ([1]), sorting and ranking ([3]), and for solving the problem of leader election [9, 12] and network initialization, i.e., the problem of assigning unique identifiers to stations in anonymous networks, [8]. More generally, the authors in [11] present an energy efficient algorithm to simulate parallel algorithms on mesh-like wireless networks. In [6], it is proved that any algorithm designed for a single-hop network in the strong-noCD model can be simulated in the weak-noCD model, with no slowdown. However, for this solution, a $O(n)$ preprocessing time is needed. Notice that both the above solution assume the knowledge of the number of stations in the network.

All the algorithms above can be divided essentially in two (or three) different classes depending on whether or not the number of stations (or an upper bound for it) is known. It is immediate that the second scenario is much more realistic. On the other hand, designing algorithms in such a scenario is a much more complicated task. In [5] the authors provide an algorithm that computes an approximation of the number of stations in the network in the strong-noCD model, in time $O(\log^{2+\epsilon} n)$ where each station spends energy $O((\log\log n)^{\epsilon})$, for any constant $\epsilon > 0$. Note that in [5], the energy is assumed to be proportional to the number of messages transmitted/listened and not to their total size. However, several messages in that protocol may have size of $\Omega(\log n)$ bits. The algorithm presented in [5] assumes no collision detection capability and outputs, with high probability, an estimation n_0 of the number of stations that satisfies the following: $n/2^6 \leq n_0 \leq 2n$. The authors in [6] claim that similar techniques can be used also in the weak-noCD model.

Our results. In this paper we present algorithms that compute the number of stations in the network in the CD and the weak-noCD models. Let n be the number of stations in the network. We show that if each station can detect collisions, in time $O(\log n \log\log n)$ the algorithm outputs an estimation m of the number of stations such that $m/2 < n < 2m$, where m denotes the output of the algorithm. In the weak-noCD model, we present an algorithm that accomplishes the same task in time $O(\log^2 n)$, maximum energy $O(\log n)$ per station and outputs an estimation m such that $m/3 < n < 3m$. Finally we show that, under the assumption that the approximate number of station is known, there exists an algorithm that computes the minimum in the weak-noCD model in time $O(\log^2 n)$. Our algorithms are randomized and the results hold with high probability (i.e., probability $1 - O(n^{-c})$ for some positive constant c). Our protocols for estimating the number of stations use only bit messages. Due to lack of space, we only outline the proofs here. Formal proofs will appear in the final version of the paper.

2 Estimating the Number of Stations

In this section we present two algorithms to estimate the number of stations in the network. We first start by describing the algorithm in the CD model. Then, we present an algorithm in the weak-noCD model.

2.1 Estimating the Number of Stations in the CD Model

In this section we present an algorithm that computes an approximation of the number of stations based on the assumption that, whenever a collision occurs, each station in the network detects it.

The basic idea of the algorithm is the following. Assume m is an estimation of the number of stations in the network. In each *phase* of the algorithm, each station may select uniformly at random and independently from the other stations, an integer r between 1 and m. At this point, the algorithm consists of m rounds. During round r, all the stations that hold a value equal to r send a message on the channel. Since stations can detect collisions, each station counts the number of "empty" slots, i.e., the number of rounds in which no station sent a message. This number is a random variable occurring in the well-known occupancy problem where k balls are randomly thrown into b bins and is known to be sharply concentrated around its expected value as the following theorem states.

Theorem 1 ([7]). *Let $r = k/b$, and let Z be the number of empty bins when k balls are thrown randomly into b bins. For $\lambda > 0$ it holds that:*

$$\mu = E(Z) = b\left(1 - \frac{1}{b}\right)^k \simeq be^{-r} \qquad \Pr\left[|Z - \mu| \geq \lambda\right] \leq 2\exp\left(-\frac{\lambda^2}{2k}\right)$$

By Theorem 1, if the estimation is close to the actual number of stations, the number of empty slots will be around m/e. So, if the number of empty bins is much higher (resp. much smaller) than m/e, the estimation m is increased (resp. decreased) and a new phase is executed. The above solution clearly needs $\Omega(n)$ rounds to terminate.

In order to avoid this unnecessary waste of time, we would like to use an algorithm in which each phase is executed by a polylogarithmic number of stations. Indeed, in this case, the number of stations is big enough to guarantee a good estimation of the actual number of stations in the network and, at the same time, the number of rounds needed to execute the algorithm is low. Notice that, from the point of view of energy consumption, this strategy also leads to low consumption since, in each phase, most of the stations will be in a sleep state. Unfortunately, the above strategy seems to require the knowledge of the number of stations.

One issue to address is the function we use to compute the new estimation of the number of stations. A first idea could be to simply double (or reduce by a half) the current estimation in order to obtain the new one. It is immediate that using this strategy, $O(\log n)$ phases will be enough in order to compute a good approximation of the number of sensors. On the other hand it is possible to reduce this number phases by choosing a function that reaches n more quickly, say in $O(\log \log n)$ phases. The problem with this second type of function is that if we compute an estimation m that is much higher than the actual value n, we may end up spending a huge number of rounds before realizing that the estimation is wrong.

What we are going to use is somehow derived by the strategy above. During each phase, a station decides to sleep or to execute the algorithm with a probability that is (roughly) inversely proportional to the current estimation of m. The reason of such relation is based on the idea that, whenever m grows, the number of stations executing each phase decreases. Using this relation between the number of active stations and the estimation, we guarantee that, with high probability each step in the algorithm is correct. On the other hand, the number of time slots in each phase is, of course, a function that is increasing as m increases. However, this function is polylogarithmic in m and this guarantees that every phase does not take too much time. This also allows us to use a "quick" way to compute a very rough estimation of n, starting from which we refine the estimation using a binary search.

We are now ready to define algorithm CountCD. We will use a simple building block, the procedure CountEmpty. This procedure takes as input the current estimation m of the number of sensors and works as follows. Let $f(m) = \alpha \log m$, for sufficiently large α. When invoked, a station switches to the sleep mode for $f(m)$ time slots with probability $1 - f(m)/m$, or stands awake and participates in the current phase with probability $f(m)/m$. If the station is awake, it selects an integer r between 1 and $f(m)$ and, during round r, it sends a message on the channel. Based on the number of empty slots, each node participating in the current phase decides whether the current estimation is higher than, lower

than, or very close to the actual number of stations. At the end of the phase, all the stations wake up and the station that first transmitted successfully during the current phase announces whether the current estimation is higher than, lower than, or very close to the actual number of stations. In case no station successfully sent a message during a given phase, we distinguish between two cases. If the participating stations have decided that the current estimation is lower than the actual number of stations then they all send a "lower" message at the last round. If a collision appears, then this is realized by all stations as a "lower" message. Otherwise, if the participating stations have decided that the current estimation is higher than the actual number of stations then no station sends any message and this silence is realized as a "higher" message.

As stated above, we divide algorithm CountCD into two epochs which we call QuickStart and SlowEnd. In the first epoch, starting with an estimation $m = 2$, each station runs the procedure CountEmpty with input the current estimation, sets the current estimation to m^2 and repeats until the call of CountEmpty on input the current estimation returns that the current estimation is higher than or very close to the number of stations. If an estimation very close to the number of stations is found, then the algorithm CountCD terminates. Otherwise, let m be the estimation which CountEmpty found to be higher than the actual number of stations. Then, the second epoch begins and executes a binary search in the set $\{i, i+1, \ldots, 2i\}$ where i is such that $\sqrt{m} = 2^i \leq n \leq 2^{2i} = m$ (i.e., $i = O(\log n)$), based on the outcomes of CountEmpty. Assuming that procedure CountEmpty correctly decides whether the current estimation is much higher, much lower, or very close to the actual number of stations, algorithm CountCD runs in $O(\log n \log\log n)$ time (i.e., $O(\log\log n)$ calls of CountEmpty in each of the two epochs QuickStart and SlowEnd).

It is clear that one of the crucial points in the algorithm is the correct estimation of the range in which the number of empty slots should belong to in order for the current estimation of the number of stations to be accepted by CountEmpty. The next lemma provides bounds on the number of empty slots in a phase depending on whether the estimation is much higher than, much lower than, or very close to the actual number of stations. Intuitively, the number of stations that are awake in this round will be much higher, much lower, or very close to $f(m)$ and the number of empty slots will be much higher, much lower, or very close to $f(m)/e$, respectively. The proof uses Theorem 1 and Chernoff bounds.

Lemma 1. *Consider a phase of algorithm* CountCD*, let m be the current estimation of the number of stations in the network and denote by O the number of empty time slots. There exist positive constants α_1, α_2 and α_3 such that the following hold:*

- *If $m \geq 2n$, then $\Pr[O < 0.55f(m)] < 2n^{-\alpha_1}$.*
- *If $m \leq n/2$, then $\Pr[O > 0.2f(m)] < 2n^{-\alpha_2}$.*
- *If $n/\sqrt{2} \leq m \leq n\sqrt{2}$, then $\Pr[O < 0.2f(m) \text{ or } O > 0.55f(m)] < 2n^{-\alpha_3}$.*

By Lemma 1, procedure CountEmpty suffices to compare the number of empty slots O in each phase with the lower and upper thresholds $0.2f(m)$ and $0.55f(m)$.

If O is higher than the upper threshold or lower than the lower threshold, then the current estimation is almost surely larger than $2n$ or smaller than $n/2$, while if O is between the two thresholds, then the current estimation is almost surely correct.

We can thus prove the following statement.

Theorem 2. *Let $n > 2$ be the number of stations in the network. With high probability, algorithm* CountCD *runs in time $O(\log n \log\log n)$ and outputs an estimation m of the number of stations such that $n/2 < m < 2n$.*

2.2 Estimating the Number of Stations without Collision Detections

The algorithm presented in the previous section exploits the collision detection capability in order to verify whether the current estimation is smaller or bigger than the actual number of stations. As stated above, during each round, each station can determine whether or not a message was sent. More specifically, when the estimation is smaller than the actual number of stations, the number of rounds in which no station sends a message is small. Conversely, when the current estimation is bigger than the number of stations, the number of rounds in which no station sends a message is high.

It could be tempting to use similar arguments in the weak no-CD model. In this model we may count the number of time slots in which a message was successfully sent. In order to use this idea, we need the following counterpart of Theorem 1 for the number of bins containing exactly one ball in the classical balls-to-bins process.

Theorem 3. *Let $r = k/b$, and let Z be the number of bins containing exactly one ball when k balls are thrown randomly into b bins. For $\lambda > 0$ it holds that:*

$$\mu = E(Z) = k\left(1 - \frac{1}{b}\right)^{k-1} \simeq ke^{-r} \qquad \Pr[|Z - \mu| \geq \lambda] \leq 2\exp\left(-\frac{\lambda^2}{2k}\right)$$

It turns out that in the case of no collision detection, it is not possible to use the QuickStart algorithm to obtain a rough estimation of the number of stations. Consider the case in which the current estimation is smaller than the number of stations. In this case, the number of collision is high and the number of slots containing exactly one message is small. Conversely, consider the case in which the estimation is bigger than the number of stations. In this case, the number of empty bins will be high and, again, the number of rounds in which a message is successfully sent is small. Since in the weak no-CD model it is not possible to distinguish between a collision and an empty slot, an algorithm cannot distinguish between the two cases.

On the other hand, whenever the current estimation is close to the number of stations, the number of rounds in which a message is successfully sent should be around $f(m)/e$. For this reason, starting from an estimation $m = 2$, whenever m is wrong, we can double it until we reach a value close to the actual number of stations. We are left to determine the thresholds to be used in the algorithm

for deciding whether or not the current estimation should be accepted. We call CountSingle the procedure that checks whether the current estimation is close to the number of stations or not. Again, each station wakes up at the end of each phase in order to realize whether or not the algorithm should continue. The station that first transmitted successfully in the current phase will announce the result at the end of the phase. Silence is realized as a "continue" message.

As described so far, we have outlined the algorithm CountnoCD which works in the strong-noCD mode. The main problem in the weak-noCD model is that the station that first transmitted successfully during a phase does not know it (since it cannot detect whether its message was successfully sent) in order to announce the result at the end of the phase. To overcome this and extend our protocol in the weak-noCD model, we can use messages of two bits in each round. The first bit is used as above while the second bit is used to encode the binary representation of the round of the first successful transmission. After i successful transmissions in the current phase (for $i \geq 1$), the second bit of the message of any transmitting node is set to the i-th least significant bit of the round in which the first node successfully transmitted.

Using Theorem 3 and Chernoff bounds we can prove the following lemma.

Lemma 2. *Consider a phase of algorithm* CountnoCD*, let m be the current estimation of the number of stations in the network and denote by O the number of time slots with successful transmissions. There are positive constants β_1 and β_2 such that the following hold:*

- *If $m \leq n/3$ or $m \geq 3n$, then* $\Pr\left[O > \frac{f(m)}{4}\right] < 2n^{-\beta_1}$.
- *If $2n/3 \leq m \leq 4n/3$, then* $\Pr\left[O < \frac{f(m)}{4}\right] < 2n^{-\beta_3}$.

By Lemma 2, procedure CountSingle suffices to compare the number O of time slots with successful transmissions in each phase with the threshold $f(m)/4$. If O is lower than the threshold, then the current estimation is almost surely wrong, while if O is higher than the threshold, then the current estimation is almost surely correct.

We can also show the following technical lemma which can be used to compute an upper bound of $O(\log n)$ on the energy of each node (i.e., total number of rounds at which a node is awake). The same bound holds for algorithm CountCD as well.

Lemma 3. *Consider the sequence of independent random variables X_i for $i = 1, ..., k$, such that $X_i \in \{0, i\}$ with $\Pr[X_i = i] = i/2^i$. Let $X = \sum_{i=1}^{k} X_i$. It holds that $\Pr[X > 6k] < 4^{-k}$.*

We can thus prove the following statement for algorithm CountnoCD.

Theorem 4. *Let $n > 2$ be the number of stations in the network. With high probability, the algorithm* CountnoCD *runs in time $O(\log^2 n)$, requires maximum energy $O(\log n)$ per node and outputs an estimation m of the number of stations such that $n/3 < m < 3n$.*

3 Computing the Minimum

We now turn to the problem of computing the minimum in the weak-noCD model. Assume each station possesses a value chosen with unknown distribution from an unknown range. We wish to design an algorithm that outputs the minimum value in the network. We assume that an upper bound on the number of stations in the network is known, otherwise we can use the algorithm presented in the previous section to estimate its value.

Let us assume that the stations hold different values. A simple idea is the following: each station randomly selects a round between 1 and n during which it sends its value on the channel. Whenever a value is successfully transmitted, all the station that hold a value bigger than the one sent, switch to the sleep state. The remaining stations continue the algorithm until a single station is alive. It is immediate that, whenever the probability distribution of the values is unknown, such an algorithm requires $O(n^2)$.

Ideally, the above algorithm tries to divide into two sets the stations so that all the station in the first set, i.e., the ones with value bigger than the received one, switch to the sleep state, while the station in the second set continue the execution. If, in each phase, we could guarantee that the set of awake stations is a constant fraction of the corresponding set in the previous phase, we could reduce the number of phases to $O(\log n)$.

To solve this problem, we will use the oversampling technique introduced in [10] that can be described as follows: from a set of n values, select equiprobably ps samples. Let $x_1, \ldots, x_{ps}$ be the sorted sequence. Consider the subsequence $x_s, x_{2s}, \ldots, x_{(p-1)s}$ and assign to the set $j = 2, \ldots, p-1$ all the values in the range $(x_{(j-1)s}, x_{js})$. All the values less than x_s will be assigned to the set with index 1 and, similarly, all the values greater than $x_{(p-1)s}$ will be assigned to the set p.

The next theorem states that the size of the sets constructed using the above technique is approximately the same with high probability:

Theorem 5 ([2]). *Let n be the number of values, let p be the number of sets*[1]*, and let s be the oversampling ratio. Then, for any $\alpha \geq 1 + 1/s$, the probability that a set contains more than $\alpha n/p$ values is at most $ne^{-(1-1/\alpha)^2 \alpha s/2}$*

Given the above theorem, we can divide the set of values held by the stations into two subsets, of approximately the same size with high probability, by using $s = O(\log n)$. As discussed above, this reduces the number of phases to $O(\log n)$.

One of the hidden assumption of Theorem 5 is the fact that the values are distinct. In order to meet this requirement, the station will randomly select in the first phase a value r in the range $\{1, \ldots, n^2\}$. A station holding a value m will broadcast the pair (m, r). We write $(m_1, r_1) < (m_2, r_2)$ iff $m_1 < m_2$ or $(m_1 = m_2$ and $r_1 < r_2)$.

[1] Notice that in [10] the value p represents the number of processors in a parallel machine. However, this restriction is immaterial as in the proof of the theorem p is used as a parameter in the selection probability.

In order to reduce the number of rounds within each phase, we need to reduce the number of stations that send a message. As in the previous sections, we will use self-selection during each phase. More specifically, let c be a constant. During phase i, for $i = 0, \ldots, \log n - \log\log n$, executes phase i with probability $2^i \log n/n$. Each awake station selects a round in the range $\{1...12ce \log n\}$ in which it will send its value on the channel. At the end of phase i, on average, each station has received $12c \log n$ values. By Theorem 5, with $\alpha = 3/4$ and $p = 2$, the probability that the number of awake stations in the subsequent phase is at least $3n/4$ is at most n^{-c}.

The algorithm works as follows: All stations are initially awake. Each station runs $3\log n - \log\log n$ phases. Each phase has $\log n$ rounds. For $i = 0, ..., \log n - \log\log n - 1$, each station which is awake at the beginning of phase i, participates to the algorithm with probability $\frac{2^i}{n}\log n$. If it participates to phase i, it equiprobably selects one of the rounds of phase i to transmit its value. For $i = \log n - \log\log n, ..., 3\log n - \log\log n$, each station which is awake at the beginning of phase i equiprobably selects one of the rounds of phase i to transmit its value. When a station hears a value smaller than its value, it becomes sleeping. The minimum value is the last transmitted value.

We will show that the algorithm correctly computes the minimum value with high probability. Let n_i be the number of awake stations at the beginning of phase i. If $n_i \leq n/2^{i+1}$, then, it is certainly $n_{i+1} \leq n/2^{i+1}$. Assume that $n/2^{i+1} < n_i \leq n/2^i$. Then, the number of successfully transmitted values in phase i is at least $\alpha \log n$ for some positive constant α. This follows by considering stations transmitting within the phase as balls and rounds as bins; then the number of successful transmissions within the phase is the number of bins receiving exactly one ball in the corresponding balls-to-bins game. The probability that more than half of the n_i stations that are awake at the beginning of phase i will still be awake at the beginning of phase $i+1$ is the probability that the transmitted values will be larger than the values stored in the $n_i/2$ stations holding the smallest values, i.e., at most $n^{-\alpha}$.

Now consider a phase of the last $2\log n$ phases. Each of the stations that are awake at the beginning of phase $\log n - \log\log n$ has constant probability (say β) of neither transmitting successfully nor sleeping during each of the next phases. So, the probability that some node is still awake after the last phase is at most $\log n \cdot \beta^{2\log n} \leq n^{-c}$ for some constant c.

Theorem 6. *There exists an algorithm that computes the minimum in the weak-noCD model in time $O(\log^2 n)$, with high probability.*

4 Conclusions

In this paper we have presented an algorithm for estimating the number of stations in an anonymous wireless network. The algorithm is based on the assumptions the station can detect collisions. The algorithm presented runs in time $O(\log n \log\log n)$, its expected energy consumption is $O(\log\log n)$ and it outputs

and estimation m such that $n/2 \leq m \leq 2n$. Furthermore we have shown that a similar technique can be used to compute in time $O(\log^2 n)$ and maximum energy $O(\log n)$ an estimation of the number of stations in the weak_noCD model. In this case the estimation m computed by the algorithm is such that $n/3 \leq m \leq 3n$. Finally we have shown that in the weak-noCD model it is possible to compute the minimum in time $O(\log^2 n)$.

References

1. R. S. Bhuvaneswaran, J. L. Bordim, J. Cui, and K. Nakano. Fundamental Protocols for Wireless Sensor Networks. In *Proc. of the 15th International Parallel and Distributed Processing Symposium (IPDPS '01)*, 2001.
2. G.E. Blelloch, C.E. Leiserson, B.M. Maggs, G.C. Plaxton, S.J. Smith and M. Zagha. An Experimental Analysis of Parallel Sorting Algorithms. *Theory of Computing Systems*, 31(2), pp. 135-167, 1998.
3. J. L. Bordim, K. Nakano, and H. Shen. Sorting on Single-Channel Wireless Sensor Networks. In *Proc. of the International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN '02)*, pp 153-158, 2002.
4. J. Elson and D. Estrin. Time Synchronization for Wireless Station Networks. In *Proc. of the 15th International Parallel and Distributed Processing Symposium (IPDPS '01)*, Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing, 2001.
5. T. Jurdzinski, M. Kutylowski, J. Zatopianski. Energy-Efficient Size Approximation of Radio Networks with No Collision Detection. In *Proc. of the 8th Annual International Conference on Computing and Combinatorics (COCOON '02)*, LNCS 2387, Springer, pp. 279-289, 2002.
6. T. Jurdzinski , M. Kutylowski , J. Zatopianski. Weak Communication in Radio Networks. In *Proc. of the 8th International Euro-Par Conference (EuroPar '02)*, LNCS 2400, Springer, pp. 965-972, 2002.
7. R. Motwani and P. Raghavan. Randomized Algorithms. *Cambridge University Press*, 1995.
8. K. Nakano, S. Olariu. Energy-Efficient Initialization Protocols for Radio Networks with No Collision Detection. In *Proc. of the 2000 International Conference on Parallel Processing (ICPP '00)*, pp. 263-270, 2000.
9. K. Nakano, S. Olariu. Randomized Leader Election Protocols in Radio Networks with No Collision Detection. In *Proc. of the 11th International Conference on Algorithms and Computation (ISAAC '00)*, LNCS 1969, Springer, pp. 362-373, 2000.
10. J.H. Reif and L.G. Valiant. A Logarithmic Time Sort for Linear Size Networks. *Journal of the ACM*, 34(1), pp. 60-75, 1987.
11. M. Singh, V. Prasanna, J. Rolim, and C. Raghavendra. Collaborative and Distributed Computation in Mesh-Like Wireless Sensor Arrays. In *Proc. of the 8th IFIP-TC6 International Conference on Personal Wireless Communications (PWC '03)*, LNCS 2775, Springer, pp. 1-11, 2003.
12. D. E. Willard. Log-logarithmic Selection Resolution Protocols in a Multiple Access Channel. *SIAM Journal on Computing*, 15, pp. 468-477, 1986.